Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Volumen III
Anexos

PROYECTO FINAL DE CARRERA

"DISEÑO DE UNA PLATAFORMA DOCENTE PARA EL APRENDIZAJE DE MICROCONTROLADORES 'PIC' DE MICROCHIP"

PFC presentado para optar al título de Ingeniería Técnica Industrial especialidad ELECTRÓNICA INDUSTRIAL
por **Víctor Bueno Álvez**

Barcelona, 12 de Enero de 2011

Tutor proyecto: Herminio Martínez García
Departamento de Ingeniería electrónica (DEEL)
Universitat Politècnica de Catalunya (UPC)

Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Anexo I
Memoria PFC1

"DISEÑO DE UNA PLATAFORMA DOCENTE PARA EL APRENDIZAJE DE MICROCONTROLADORES 'PIC' DE MICROCHIP"

PFC presentado para optar al título de Ingeniero Técnico Industrial especialidad ELECTRÓNICA INDUSTRIAL

por **Víctor Bueno Álvez**

Barcelona, 12 de Enero de 2011

Tutor proyecto: Herminio Martínez García
Departamento de Ingeniería electrónica (DEEL)
Universitat Politècnica de Catalunya (UPC)

ÍNDICE MEMÓRIA

RESUMEN

En el documento que a continuación se muestra se encuentran representados los aspectos más relevantes a la hora de realizar una placa entrenadora de microcontroladores. Los temas están clasificados de forma que se facilite la comprensión de los diferentes problemas de esta tarea, con una introducción de cada capítulo y las soluciones escogidas así como las posibles alternativas. Para finalizar el documento, se han añadido una lista de prácticas con los enunciados y programas correspondientes para experimentar con los diferentes periféricos de la placa entrenadora.

RESUM

En el document que a continuació es mostra es troben representats els aspectes més rellevants a l'hora de realitzar una placa entrenadora de microcontroladors. Els temes estan classificats de forma que es faciliti la comprensió dels diferents problemes d'aquesta tasca, amb una introducció de cada capítol i les solucions escollides així com les possibles alternatives.

Per finalitzar el document, s'han afegit un llista de pràctiques amb els enunciats i programes corresponents per tal d'experimentar amb els diferents perifèrics de la placa entrenadora.

ABSTRACT

In the document that shown below are represented all the important aspects to do the design of a microcontroller trainer board. The topics are classified so as to facilitate comprehension of the various problems of this task, with an introduction of each chapter and the solutions adopted and possible alternatives.

To finalize the document, have added a list of practice statements and programs to experiment with the peripherals of the microcontroller trainer board.

CAPÍTULO 1: INTRODUCCIÓN

Con la realización de éste proyecto se pretende diseñar una placa de circuito impreso con todo lo necesario para estudiar el funcionamiento del microcontrolador PIC16F877A, así como de otros microcontroladores con patillaje compatible con las modificaciones software correspondientes. Por lo tanto, se desea diseñar un "kit" completo formado por la placa entrenadora y un conjunto de prácticas diseñadas específicamente para la placa en cuestión que permita seguir paso a paso una evolución a través de los diferentes periféricos que forman la placa. Además, ésta dispone de puertos accesibles desde el exterior, con lo cual su utilidad no queda limitada a las prácticas ofrecidas, sino que el usuario puede añadir periféricos exteriores para así poder comprobar el funcionamiento de proyectos propios, con la seguridad del correcto funcionamiento de todos los componentes conectados en la tarjeta.

1.1. Motivación

La elección de la realización de éste proyecto reside principalmente en el interés del alumno en el aprendizaje sobre los microcontroladores PIC, una familia de microncontroladores que crece con el paso del tiempo, no sólo en el número de ventas sino también en la gama de microcontroladores ofertados ampliando así el rango de aplicación de éstos.

1.2. Objetivos PFC1

La realización de éste informe resume todo el trabajo hecho hasta el momento en el proyecto, y aun no siendo un informe definitivo, trata de dar una idea acerca del alcance del proyecto final, así como una aproximación al aspecto final. En la realización de este anteproyecto se han tenido que fijar

una serie de objetivos prioritarios en esta etapa del proyecto, para no profundizar demasiado en algunos aspectos dejando de banda otros. Por lo tanto, se deja el análisis detallado para la segunda parte del proyecto.

Uno de los objetivos primordiales es el de definir el alcance del proyecto. Éste aspecto que parece tan trivial es el primero que debe ser estudiado para optimizar al máximo el tiempo, cosa que se consigue teniendo unas ideas claras acerca del proyecto. Primeramente se debe hacer un estudio del arte acerca de la materia tratada en el proyecto, y delimitar los aspectos de éste. De ésta forma, se han estudiado las diferentes propuestas de entrenadores en el mercado y se han escogido los periféricos específicos en el diseño de éste.

Una vez analizado el contexto del proyecto, se procede a la documentación sobre los diferentes aspectos para conseguir así una base donde ir desarrollando ideas acerca de los diferentes aspectos que definirán el proyecto final. Después de documentarse, decidir el camino a seguir y plantear los diferentes matices junto con el profesor, se elabora el informe presente en el que se intentan recoger los elementos principales trabajados durante ésta primera parte.

Hay que tener en cuenta, que como se ha dicho, éste anteproyecto sirve para aclarar las ideas y enfocar el trabajo a realizar durante la segunda parte de éste, por lo tanto no se considera como un informe definitivo sino como orientativo, pudiéndose producir cambios en su contenido en caso de encontrar posibles diferencias o problemas observados al llevar todos los estudios al montaje práctico.

Las principales tareas realizadas en éste etapa del proyecto, en orden cronológico, han sido:

1. Delimitación, junto al profesor, de los aspectos que debe cubrir el 'kit' (periféricos y prácticas)
2. Estudio del arte de las diferentes placas existentes
3. Estudio de los microcontroladores PIC
4. Elección del microcontrolador PIC16F877A
5. Delimitación, junto al profesor, de los aspectos que debe cubrir la placa
6. Elaboración conjunto de prácticas
7. Elaboración del informe

El conjunto de prácticas se encuentra en elaboración y por lo tanto sólo se han incluido algunas de las prácticas que ya están probadas tanto a nivel software como hardware. En el informe final éstas se encontraran recogidas en un anexo independiente con su enunciado completo y el programa.

A continuación se presenta un diagrama de Gantt representativo con el trabajo realizando durante el PFC1, así como una previsión de carga de trabajo del PFC2 y una breve propuesta de trabajo también para el período vacacional. A falta de ser definitivo, el diagrama de Gantt da una idea acerca de la carga de trabajo, y queda pendiente el estudio conjunto junto al profesor para una repartición más exacta.

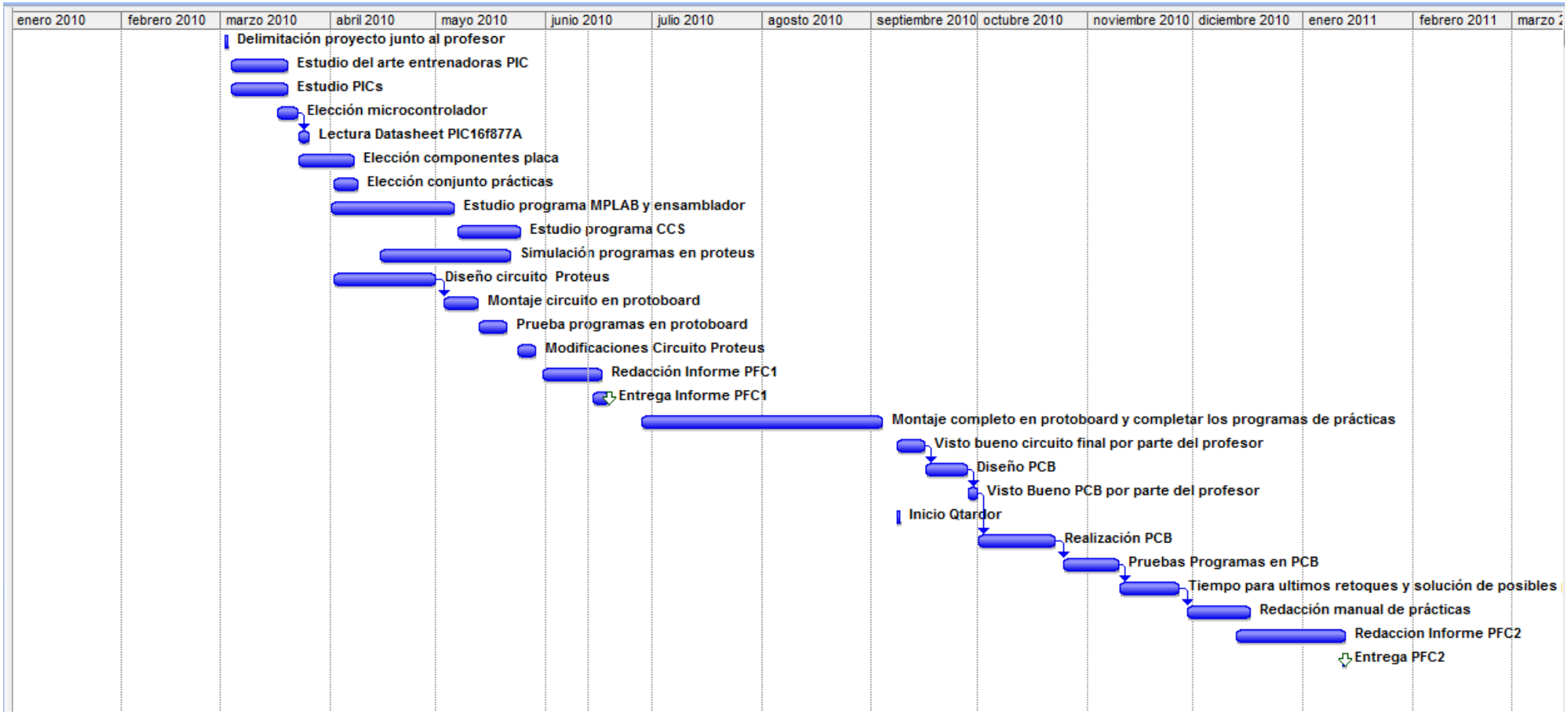


Figura 1. *Diagrama Gantt PFC*

1.3. Entrenadoras microcontroladores

Un entrenador para Microcontroladores es un laboratorio para el desarrollo de diseños tanto a nivel profesional como educativo. Es una herramienta didáctica que permite aprender la mayor parte de las tecnologías electrónicas modernas, pudiendo ser sus usuarios aficionados y estudiantes que desean formarse de la única forma efectiva que admite la electrónica: con la *PRÁCTICA*. Por otro lado, también puede ser muy interesante para ingenieros y profesionales que desean desarrollar rápidamente sus proyectos.

Una entrenadora de microcontroladores es , en esencia, un microcontrolador específico o una variedad de ellos, conectados a una serie de periféricos externos que permiten simular las diferentes operaciones a realizar con el microcontroladores, de ésta forma se podrá interactuar con el microcontrolador introduciéndole información a través de los periféricos de entrada (pulsadores, teclados, entradas analógicas,…) y observando la respuesta mediante periféricos de salida (LCD, displays 7 segmentos, Leds,…). Además, se suelen incorporar medios de comunicación (Rs-232, CAN, USB,…) así como un circuito de programación del microcontrolador que evita tener que sacar éste de la placa cuando se necesite programarlo.

1.4. Fabricación entrenadora

A la hora de decidirse a diseñar una placa entrenadora lo primero que se debe hacer es mirar que se puede encontrar en el mercado, para poder así observar hacia donde se dirigen las tendencias. En la actualidad se pueden encontrar una gran variedad de éste tipo de entrenadores desde los más avanzados con los periféricos más novedosos hasta los más sencillos y económicos para usuarios que busquen un conjunto de periféricos más sencillos. A continuación se nombrarán algunos de los entrenadores comerciales más importantes por si el lector desea profundizar algo más en alguna de las características de éstos entrenadores.

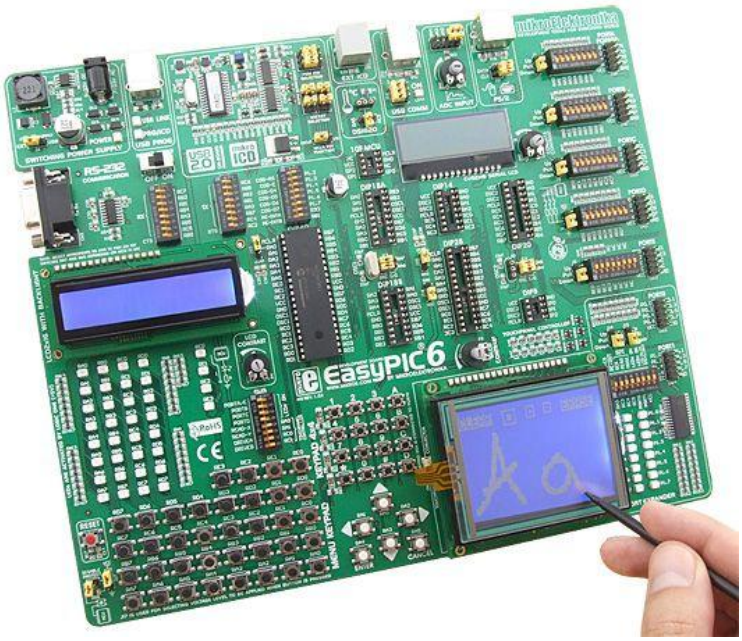


***Figura 2**. EasyPic6 - Mikroelektronika*

***Figura 3.** PIC'School - MSE*

Figura 4. *MicroPIC Trainer - MSE*

Por otro lado, también se puede encontrar por la red gran cantidad de "Hardware libre", donde algunos usuarios han diseñado su propio entrenador y ofrecen al público los esquemas, listados de componentes e incluso en algunos casos un manual de prácticas, de forma totalmente gratuita. También cabe la posibilidad de que se desee realizar un entrenador propio con unos periféricos concretos para las necesidades particulares, con éste objetivo principal se ha diseñado el entrenador de éste proyecto.

A continuación se hará un breve resumen sobre las principales características que puede incluir una placa entrenadora, donde en la placa diseñada en éstas páginas se han incluido las que se consideraban más importantes integradas en la placa dejando en manos del lector añadir alguna otra deseada a través de los puertos con conexión externa.

1. Fuente de alimentación de tensión regulada

- Transformador y puente rectificador incorporado.
- Interruptor de encendido.
- Tensión doble (5+12V)

2. Periféricos de Salida

- Display de 7 segmentos
- Entrada BCD
- Entrada Binaria
- Display LCD
- Buzzer activo.
- Leds

3. Periféricos de Entrada

- Pulsadores.
- Interruptores.
- Teclado matricial.
- Entradas Analógicas (sensores, potenciómetros)
- Generador de onda cuadrada

4. Comunicaciones

- Interfaz RS232.
- USB.
- CAN.
- Programación In-Circuit (ICSP)

CAPÍTULO 2: MICROCONTROLADORES PIC

Los microncontroladores "PIC" son una familia del tipo RISC (Reduced Instruction Set Computer), una arquitectura computacional que se basa en instrucciones cortas, esto hace que los microcontroladores PIC sean unos computadores rápidos y eficaces.

Son fabricados por TechnologyInc. y provienen del PIC1650 desarrollado originalmente por la división de microelectrónica de General Instruments. El nombre completo es PICmicro, aunque generalmente se utiliza como acrónimo de Peripheral Interface Controller.

Los microcontroladores PIC son muy utilizados tanto por aficionados como profesionales, debido esto principalmente al bajo coste, facilidad de manejo y programación, además de sus buenas prestaciones disponiendo de memoria de gran capacidad (RAM, EEPROM, FLASH,...) y una gran cantidad de dispositivos periféricos integrados, como pueden ser módulos PWM, osciladores internos, convertidores A/D así como módulos de comunicación (USART, SPI, I2C, CAN, USB...).

2.1. Características generales

A continuación se hará un breve repaso sobre algunas de las características más importantes de éste tipo de microcontroladores para ir así familiarizándonos con éstos.

Arquitectura

La arquitectura del procesador sigue el modelo Harvard, lo que implica que la CPU se conecta de forma independiente y con buses diferentes con la memoria de instrucciones y la de datos, lo cual permite ala CPU acceder simultáneamente a las dos memorias.

Además éstos microcontroladores poseen una arquitectura basada en banco de registros lo que implica que todos los componentes del sistema (puertos de E/S, temporizadores, posiciones de memoria, etc...) se encuentran implementados físicamente como registros.

Segmentación

En éstos microcontroladores se aplica la técnica de segmentación "pipe line" en la ejecución de las instrucciones. La segmentación permite al procesador realizar al mismo tiempo la ejecución de una instrucción y la búsqueda del código de instrucción siguiente. Con éste procedimiento conseguimos ejecutar cada instrucción en un ciclo, lo que equivale a cuatro ciclos de reloj. Esto no es así en las instrucciones de salto que ocupan dos ciclos ya que no se sabe la dirección de la siguiente hasta que se haya completado la operación.

Juego y formato de instrucciones

Como se ha dicho anteriormente éstos disponen de procesadores RISC, lo que indica que disponen de un repertorio reducido de instrucciones. Los modelos de gama baja disponen de 33 instrucciones, siendo 35 para los de gama media y próximas a las 60 los de gama alta.

En cuanto al formato cabe remarcar que todas las instrucciones tienen la misma longitud, 12 bits los de gama baja, 14 bits los de gama media y el valor aumenta en los de gama alta, aunque con diferencias entre ellos. Al tener todos los procesadores de la misma gama instrucciones con misma longitud facilita la optimización de la memoria de instrucciones y la construcción de programas ensambladores y compiladores.

Otra característica de las instrucciones de estos microcontroladores es que todas son ortogonales, es decir, cualquier instrucción puede manejar cualquier elemento de la arquitectura como fuente o destino.

Los microcontroladores PIC actuales incorporan una amplia gama de mejores hardware, donde se podría destacar:

- Núcleos de CPU de 8, 16 o 32 bits con Arquitectura Harvard modificada
- Memoria Flash y RAM disponible desde 256 bytes a 256 kilobytes
- Puertos de E / S (típicamente de 0 a 5,5 voltios)
- Temporizadores de 8, 16 o 32 bits
- Tecnología Nanowatt por modos de control de energía
- Periféricos serie síncronos y asíncronos: USART, AUSART, EUSART
- Convertidores analógico / digital de 8-10-12 bits
- Comparador de tensión
- Módulos de captura y comparación PWM
- Controladores LCD
- Periférico MSSP para comunicaciones Y ² C, SPI, e I ² S
- Memoria EEPROM interna con duración de hasta un millón de ciclos de lectura / escritura
- Periféricos de control de motores
- Soporte de interfaz USB
- Soporte de controlador Ethernet
- Apoyo de los controladores CAN
- Soporte de controlador LIN
- Soporte de controlador IrDA

2.2. Familias de microcontroladores PIC

Cuando se comienza a trabajar con microcontroladores PIC lo primero que se observa es que existen una gran diversidad, y a veces es difícil conocer cuál es el ideal para una aplicación determinada. Para simplificar un poco la elección se podría decir que posiblemente, la mejor manera de clasificarlos sea en función del número de bits con que trabaja, de ésta forma tenemos microcontroladores de 8, 16 y 32 bits.

Se podría pensar que la existencia de microcontroladores de 8 bits podría ponerse en cuestión, ya que, claramente, las prestaciones de los microcontroladores de 16 y 32 bits son superiores. Pero todo lo contrario, en éste momento los microcontroladores de 8 bit se encuentran dominando el mercado, debido principalmente, a que éstos "pequeños" microcontroladores son útiles y sobradamente aptos para la mayoría de las aplicaciones lo que hace absurdo utilizar microcontroladores con mayores prestaciones lo que haría incrementar el precio de la aplicación sin un aumento de sus prestaciones.

Dentro de la familia de 8 bits podemos encontrar deferentes familias que van de menores a mayores prestaciones, ésta son la serie 12, la 16 y la 18 además de otras algo menos utilizadas y algunas en desuso.

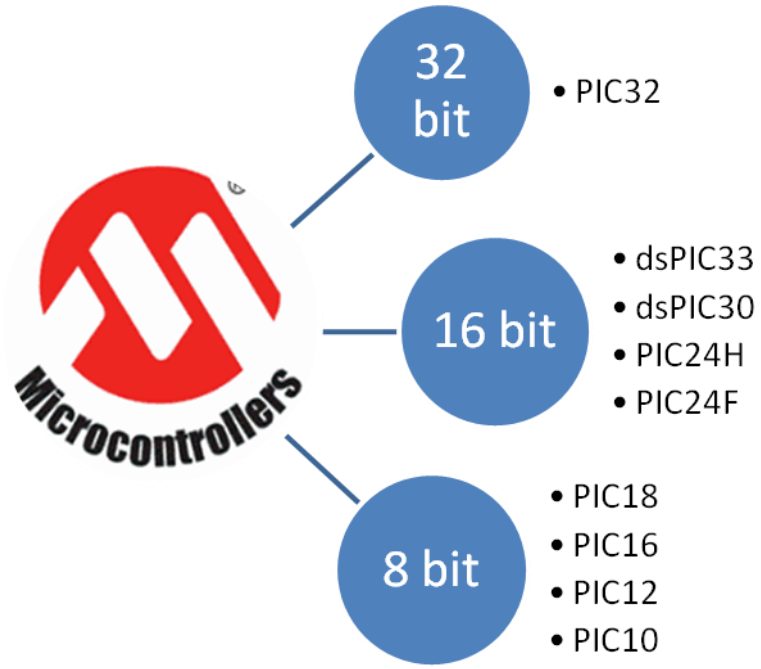


Figura 5. *Familias de microcontroladores PIC*

Dentro de todo el repertorio de microcontroladores disponibles se ha centrado la elección del microcontrolador en las dos familias más avanzadas de microcontroladores de 8 bits, la 16 y la 18, y más concretamente en los microcontroladores disponibles en encapsulado de 40 pines, ya que esto nos proporciona más puertos de E/S, cosa indispensable en esta aplicación.

Nada mejor para observar las diferentes familias de 8 bits que un resumen de la tabla proporcionada por la web de microchip (www.microchip.com) una excelente página con gran cantidad de información útil para trabajar con estos microcontroladores.

Tabla 1. *PIC 8 bits*

Arquitectura	'Baseline'	'Midrange'	'Enhanced midrange'	PIC18
Nº Pins	6-40	8-64	8-64	18-100
Funcionamiento	5 MIPS	5 MIPS	8 MIPS	10-16 MIPS
Instrucciones	33, instrucciones 12 bits	35, instrucciones 14 bits	49, instrucciones 14 bits	75, instrucciones 16 bits
Memoria de programa	Más de 3 KB	Más de 14 KB	Más de 56 KB	Más de 128 KB
Memoria de datos	Más de 138 Bytes	Más de 368 Bytes	Más de 4 KB	Más de 4 KB
Características principales	Bajos coste Fácil de aprender y usar	Buena relación calidad/precio Periféricos integrados (I2C, ADC,..)	Optimizados para código C Mapa de memoria simplificado	Multiplicador Hardware Periféricos avanzados (USB, CAN,..)
Familias	PIC10, PIC12, PIC16	PIC2, PIC16	PIC12F1xxx, PIC16F1xxx	

Atendiendo a la compatibilidad de las líneas de VSS, VDD, MCLR/Vpp, OSC1/CLKIN, OSC2/CLKOUT, se puede diferenciar entre cuatro grupos de PIC, los PIC16CXXX, PIC16FXXX, PIC18CXXX, PIC18FXXX. Estos modelos en concreto disponen de un sistema de gestión de interrupciones vectorizadas muy potente.

Tabla 2. *PIC serie 16Cxxx*

PIC16C64A	PIC16C662
PIC16CR64	PIC16C74
PIC16C65	PIC16C74A
PIC16C65A	PIC16C74B
PIC16C65B	PIC16C77
PIC16CR65	PIC16C765
PIC16C67	PIC16C774

Tabla 3. *PIC serie 16Fxxx*

Subfamilia PIC16F8x	Subfamilia PIC16F87X
PIC16F84	PIC16F77
PIC16F84A	PIC16F877A
PIC16F88	PIC16F877
Respectivas mejoras	
PIC16F77	PIC16F886/887

Tabla 4. *PIC serie 18xxxx*

PIC18C442
PIC18F452
PIC18F2455
PIC18F2550
PIC18F4550

CAPÍTULO 3:PIC 16F877A

3.1. Motivos de su elección

En un principio se planteó la elección entre dos microcontroladores de familias diferentes. Por un lado se tenía el PIC 16F877A explicado anteriormente, un microcontrolador avanzado de la serie 16 con una gran variedad de periféricos incorporados y un juego de instrucciones bastante completo. Por otro lado, estaba el PIC 18F4550, un controlador de gama alta de la serie 18 con un juego de instrucciones más amplio y algunos periféricos extra como la comunicación USB, bus CAN,…

Finalmente, se decidió utilizar el PIC16F877A para la realización de éste proyecto a consecuencia, principalmente de la mayor facilidad de programación en asembler así como la mayor disponibilidad de información acerca del funcionamiento del microcontrolador además de programas ejemplo y tutoriales de ayuda, un aspecto éste, muy importante en un proyecto didácticos como éste, donde se busca una finalidad educativa, y es destinado a ser utilizado por otros alumnos.

3.2. Características principales

El modelo 16F877A posee varias características que hacen a este microcontrolador un dispositivo muy versátil, eficiente y práctico para ser empleado en la aplicación que se pretende en éste proyecto.

Algunas de estas características se muestran a continuación:

- Soporta modo de comunicación serial, posee dos pines para ello.
- Amplia memoria para datos y programa.
- Memoria reprogramable: La memoria en este PIC es la que se denomina FLASH; este tipo de memoria se puede borrar electrónicamente (esto corresponde a la "F" en el modelo).
- Set de instrucciones reducido (tipo RISC), pero con las instrucciones necesarias para facilitar su manejo.

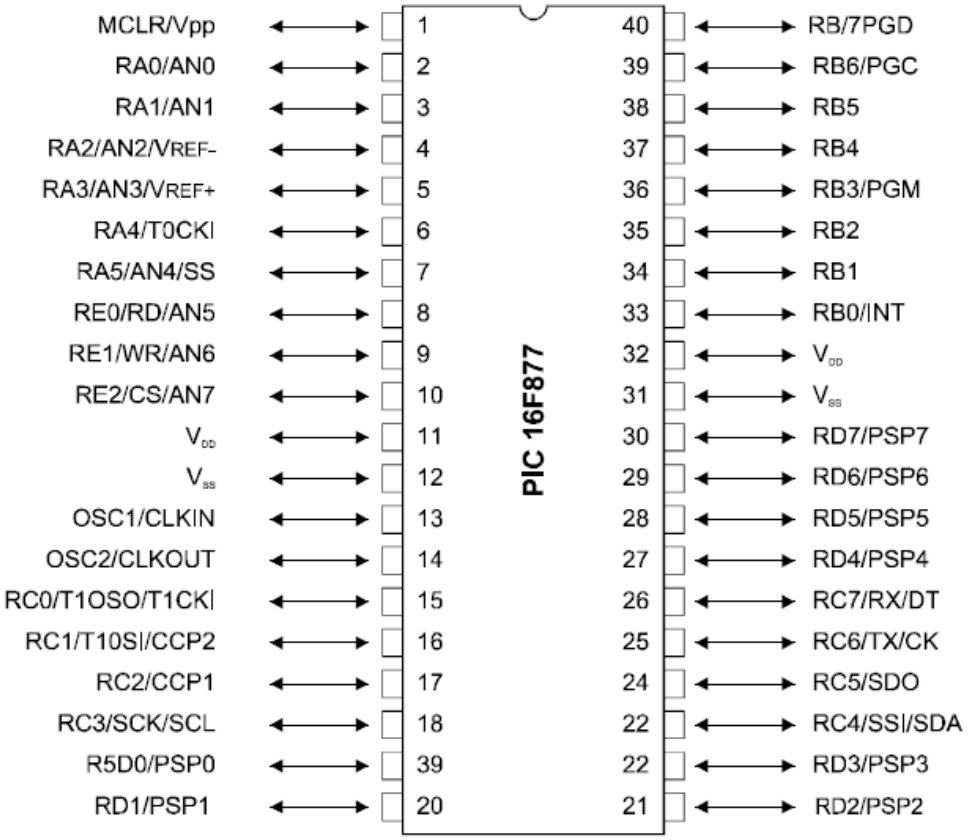


***Figura 6**. Disposición patillaje PIC16F877A*

CARACTERISTICAS

En la siguiente tabla se pueden observar las características más relevantes:

Tabla 5. *Características PIC16F877A*

	CARACTERÍSTICAS	16F877A
PROCESADOR	Frecuencia máxima	DX-20MHz
	Arquitectura	Harvard
	CPU	RISC
	Interrupciones	14 fuentes posibles
	Juego de instrucciones	35 Instrucciones de 14 bits
	Reset	Master Clear, Borwn Out, Watchdog
	Reloj	0-20 MHz
MEMORIA	Memoria de programa flash	8KB palabras de 14 bits
	Memoria de datos RAM	368 registros de 8 bits
	Memoria de datos EEPROM	256 registros de 8 bits
	Pila	8 palabras de 13 bits
	Memoria de datos externa EEPROM	Hasta 256 KBytes
PERIFERICOS	Puertos programables de E/S (A,B,C,D,E)	Hasta 33 bits, pueden ser utilizados por otros periféricos
	Número de pines	40
	Timers/Counters	2 de 8 bits y uno de 16
	Módulos Captura/comparación datos	2 de 8 bits
	Puerto serie síncrono	Configurable en modo SPI e I2C
	USART	Para conexiones RS 232
	Parallel Slave Port	8 bits + 3 bits de control
	Líneas de entrada de CAD de 10 bits	8 entradas multiplexadas
	Canales Pwm	2 de 8 bits

Tabla 6. *Descripción de los puertos*

Puerto	Nºpines	Características
A	6	RA0 /AN0 RA1/AN1 RA2, AN2 y Vref- RA3, AN3 y Vref+ RA4 (Salida en colector abierto) y T0CKI(Entrada de reloj del modulo Timer0) RA5, AN4 y SS (Selección esclavo para el puerto serie síncrono)
B	8	Resistencias pull-up programables RB0 Interrupción externa RB4-7 Interrupción por cambio de flanco RB5-RB7 y RB3 programación y debugger in circuit
C	8	RC0, T1OSO (Timer1 salida oscilador) y T1CKI (Entrada de reloj del modulo Timer1). RC1-RC2- PWM/COMP/CAPT RC1/T1OSI (entrada osc timer1) RC3-4- IIC RC3-5- SPI RC6-7 è USART
D	8	Bus de datos en PPS (Puerto paralelo esclavo)
E	3	RE0, AN5 y Read de PPS RE1, AN6 y Write de PPS RE2, AN7 y CS de PPS

Dispositivos periféricos:

- Timer0: Temporizador-contador de 8 bits con preescaler de 8 bits
- Timer1: Temporizador-contador de 16 bits con preescaler que puede incrementarse en modo sleep de forma externa por un cristal/clock.
- Timer2: Temporizador-contador de 8 bits con preescaler y postescaler.
- Dos módulos de Captura, Comparación, PWM (Modulación de Anchura de Impulsos).
- Conversor A/D de 1 0 bits.
- Puerto Serie Síncrono Master (MSSP) con SPI e I2C (Master/Slave).
- USART/SCI (Universal Syncheronus Asynchronous Receiver Transmitter) con 9 bit.
- Puerta Paralela Esclava (PSP) solo en encapsulados con 40 pines

CAPÍTULO 4: DISEÑO DE LA PLACA

4.1. Definición de la placa

Características placa entrenadora:

- Fuente de alimentación estabilizada de +5V/12V y 1,5A
- 8 salidas digitales a Led.
- 6 entradas digitales mediante pulsadores.
- 1 entrada analógica a través de un sensor de temperatura LM35.
- 2 entradas analógicas a través de potenciómetros.
- 1 teclado matricial 4x4.
- 1 módulo LCD de 2x16.
- 4 displays 7 segmentos con transistores de conmutación y controlador 4511.
- 1 reloj en tiempo real (RTC) en bus I2C modelo DS1307.
- 1 convertidor AD/DA en bus I2C modelo 8591.

4.2. Diagrama de bloques

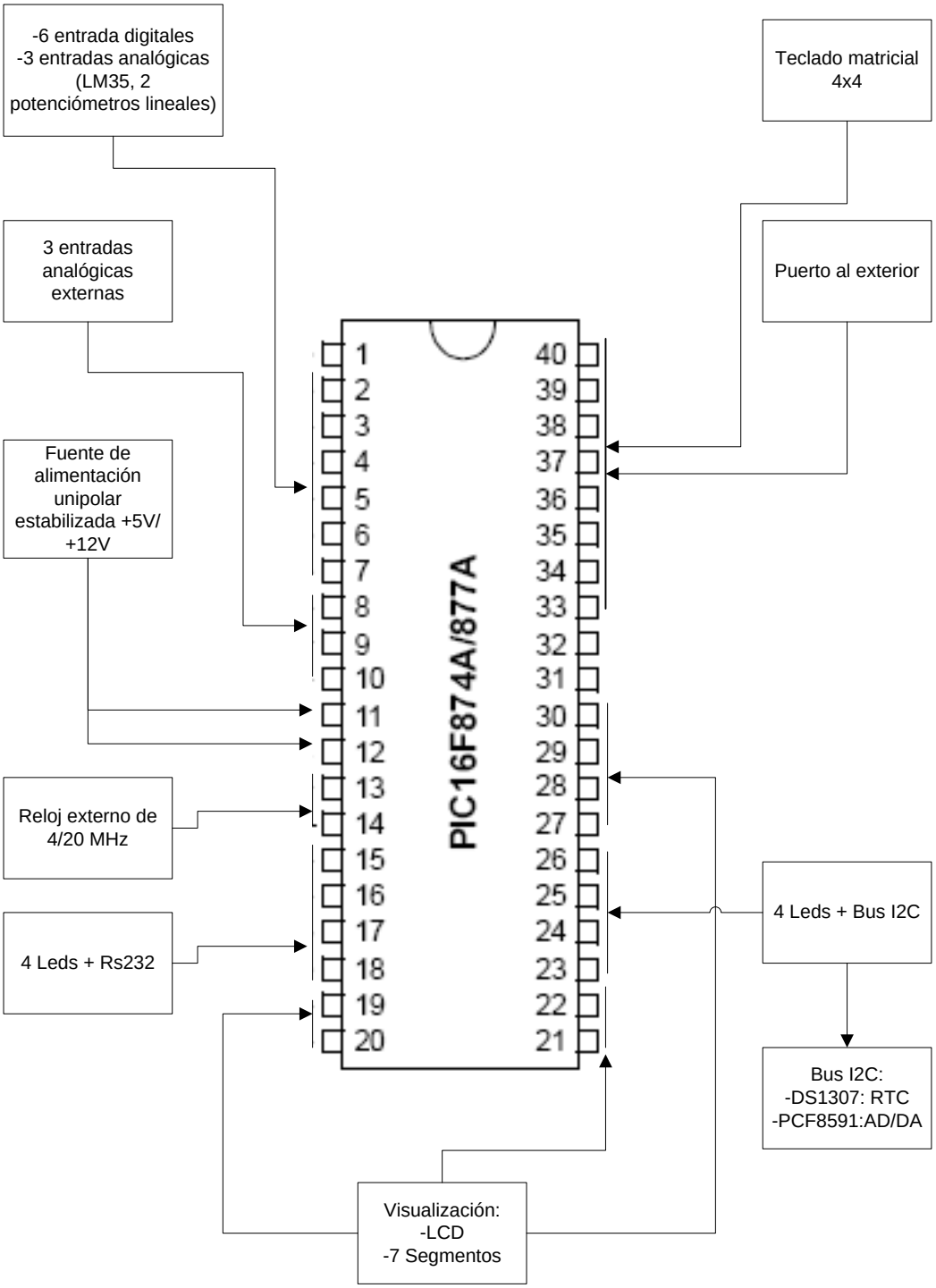


Figura 7. *Diagrama de bloques entrenador*

4.3. Definición Bloques

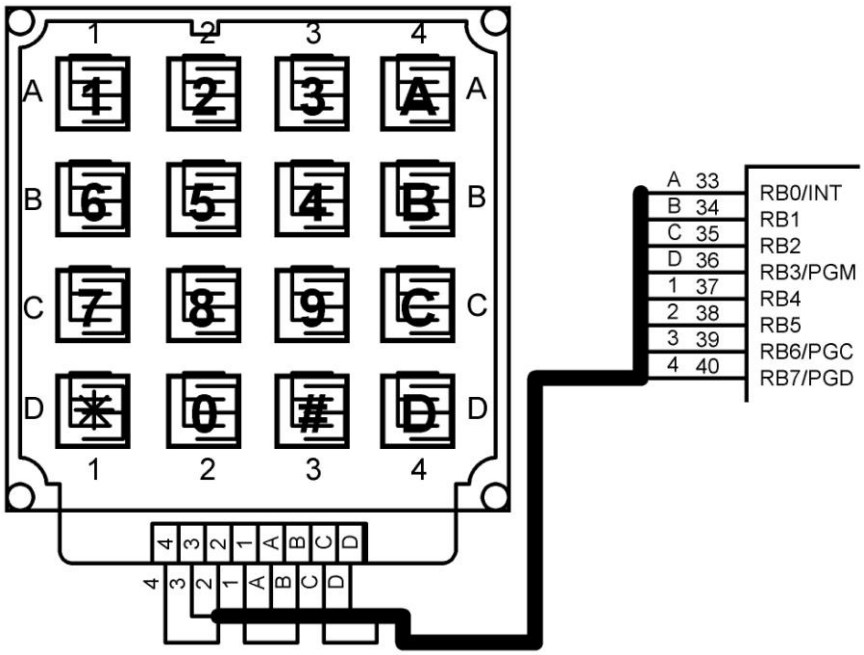


***Figura 8**. Teclado matricial 4x4*

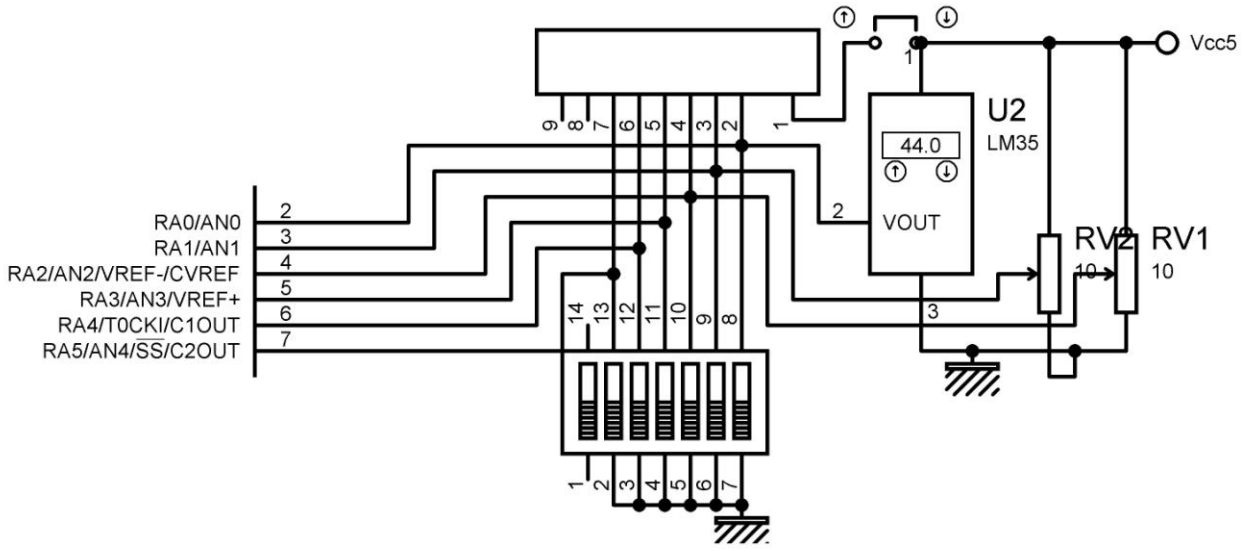


***Figura 9.** Entradas digitales/analógicas*

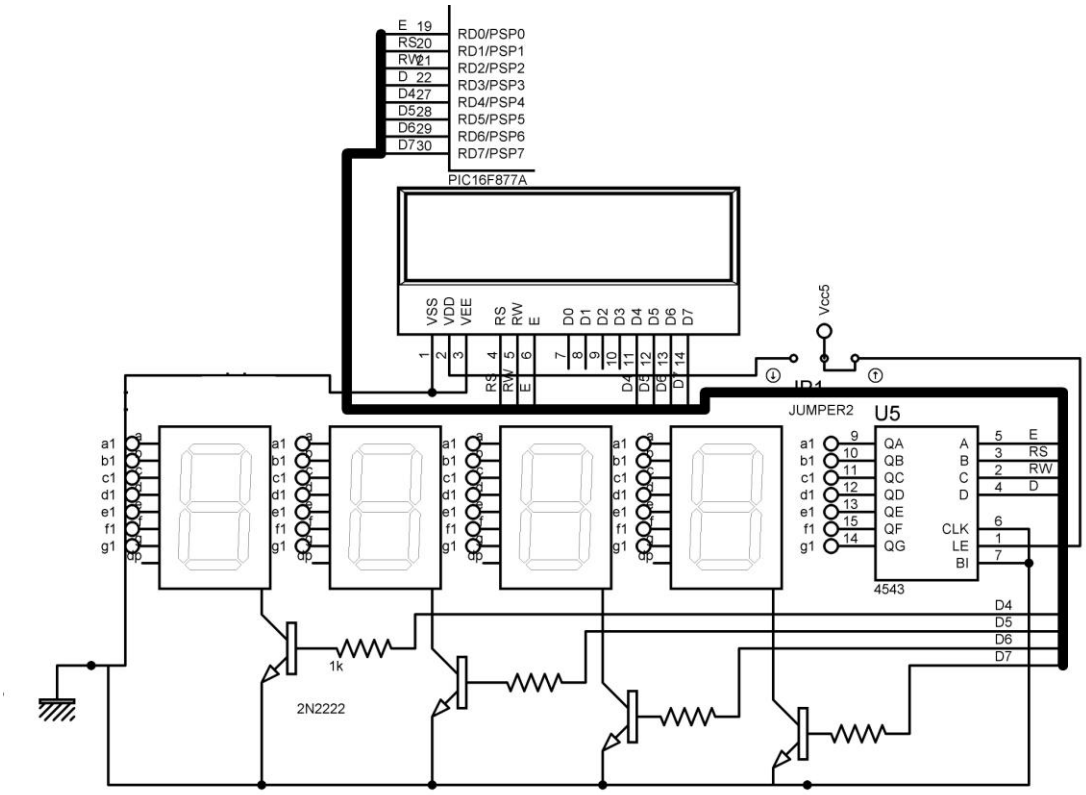


***Figura 10**. Visualización (7 segementos y LCD)*

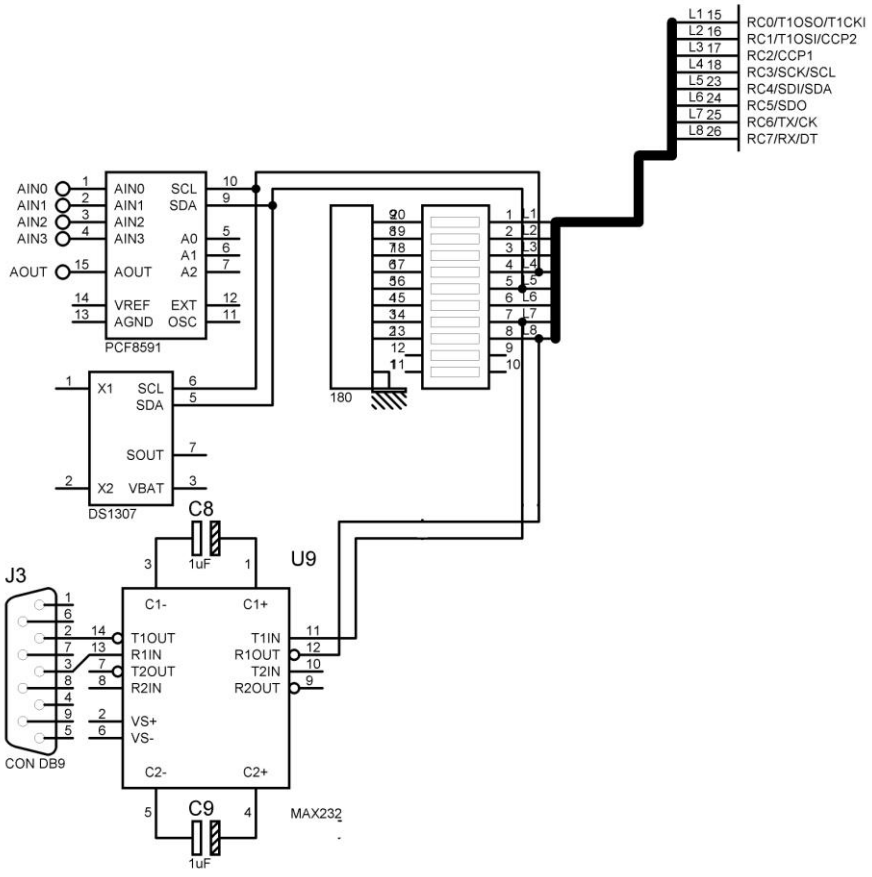


***Figura 11.** Salidas digitales y comunicaciones RS232 e I2C*

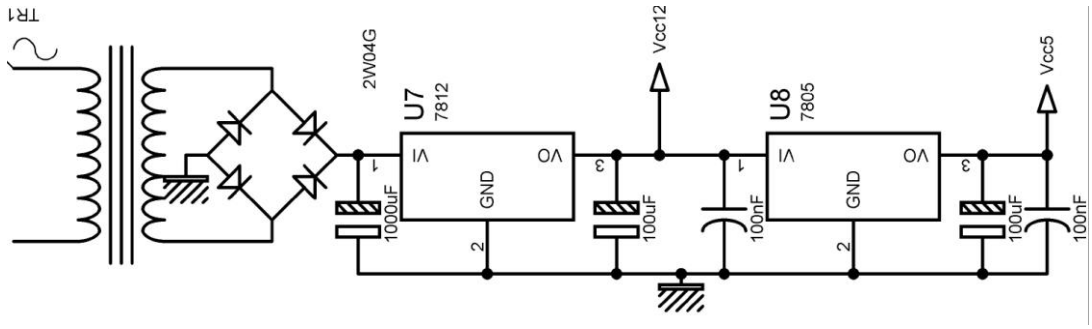


Figura 12. *Fuente de alimentación*

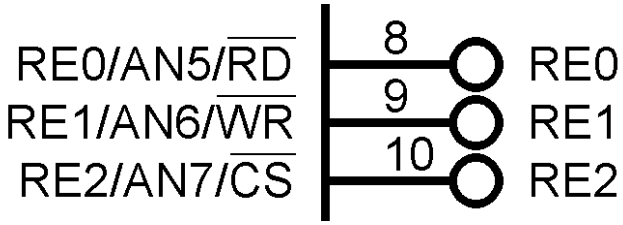


Figura 13. *Puerto E al exterior*

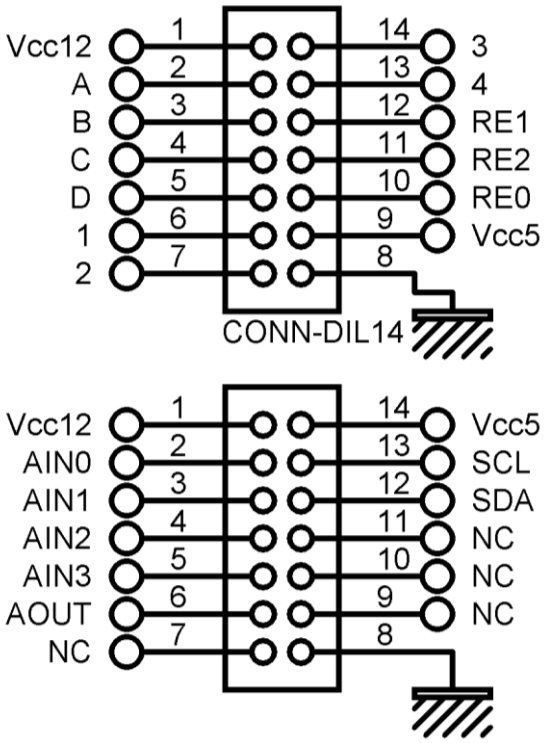


Figura 14. *Conectores externos*

Módulos externos

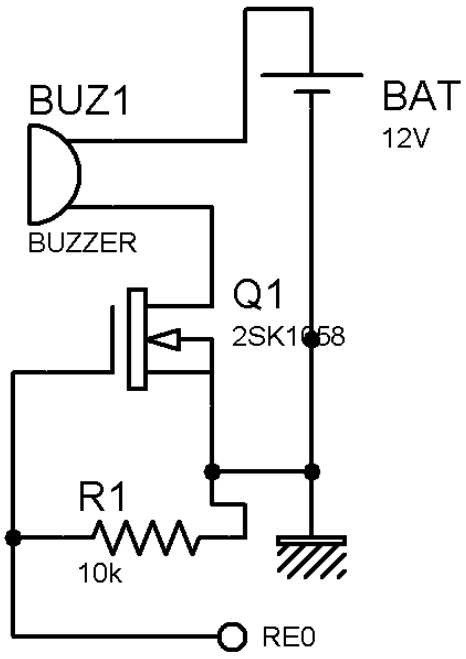


Figura 15. *Placa externa altavoz*

4.4. Circuito Final

En la siguiente figura se muestra el circuito correspondiente a la placa entrenadora diseñada. De momento este circuito puede verse modificado por cambios en caso de observar problemas prácticos al ir montando el circuito en protoboard. En éste apartado se incluirá también la PCB del circuito una vez comprobado todo el funcionamiento de éste y diseñada la PCB.

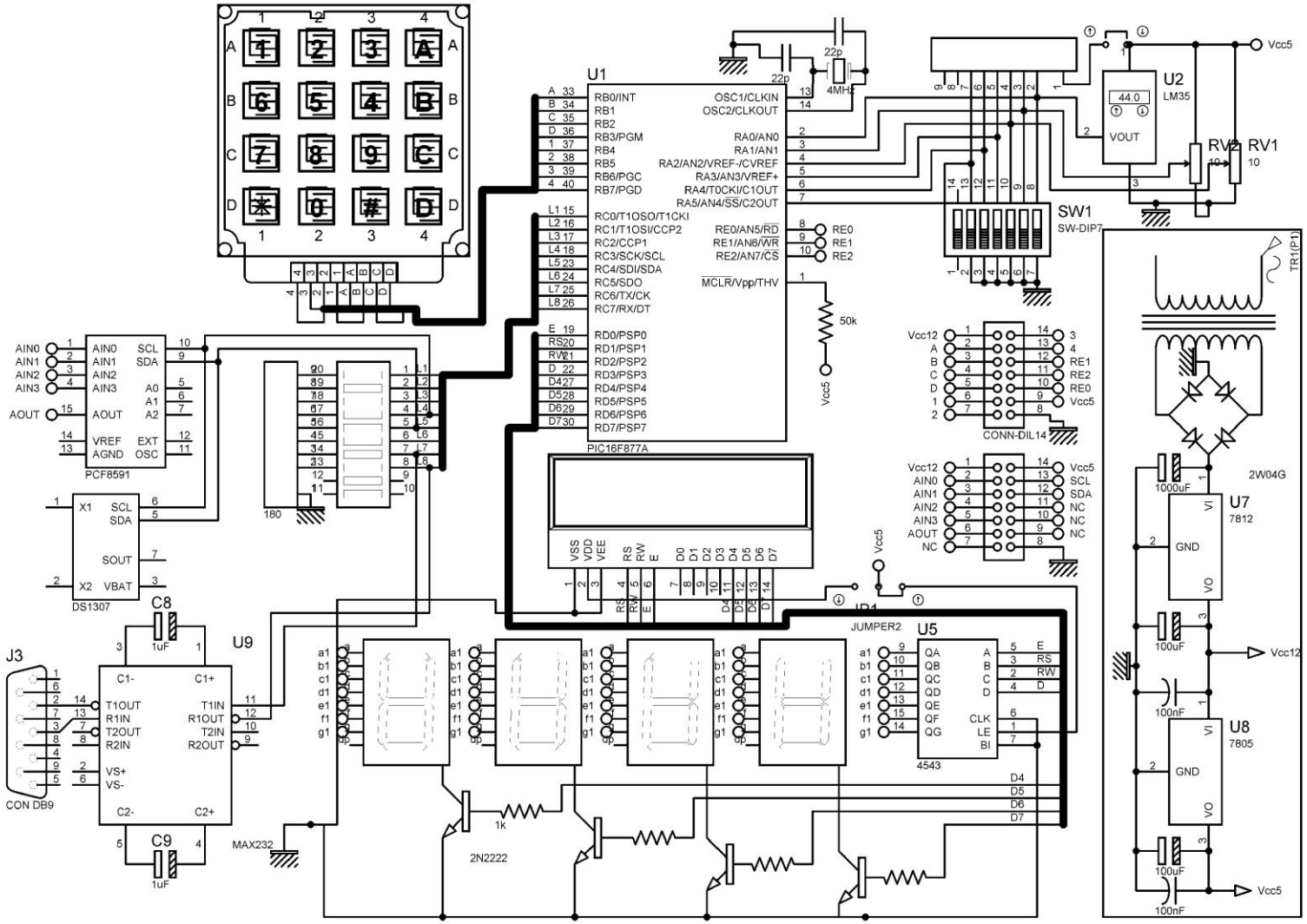


Figura 16. *Circuito Final*

4.5. Elección componentes

En éste apartado se pretende realizar un listado con todos los componentes utilizados en el montaje del circuito para así facilitar el montaje al lector.

Tabla 7. *Listado de componentes*

Componente	Cantidad
Microcontrolador PIC16F877A	1
Teclado matricial 4x4	1
LCD 16x2	1
LED	8
Pulsador	6
LM35	1
PCF8591	1
DS1307	1
MAX232	1
Display 7 segmentos	4
4543	1
7905	1
7912	1
Cristal 4MHz	1
Cristal 20MHz	1
Conector DIL14	2
Conector DB9	1
Pasivos	
Resistencias	
Potenciómetros	
Condensadores	
Zócalos	
Jumper	

CAPÍTULO 5: PRÁCTICAS DE LABORATORIO

Hasta el momento nos hemos dedicado al estudio del hardware de la placa entrenadora. Por otro lado, debemos tener en cuenta otro aspecto muy importante a la hora del diseño de éste "kit" didáctico, el conjunto de prácticas con el programa para el microcontrolador y la explicación de la práctica oportuna.

El kit está formado por una serie de 12 prácticas en las que se trabajan los distintos periféricos de la tarjeta así como las distintas operaciones que nos permite el microcontrolador. Estas se encuentran organizadas en orden ascendente de dificultad para así hacer más fácil la familiarización con el microcontrolador. Además, podríamos dividir éstas en dos grupos bien diferenciados según el lenguaje de programación con el que se trabaja. Las primeras, más sencillas, están realizadas en ensamblador mediante el programa MPLAB IDE v8.36 proporcionado gratuitamente por microchip. El lenguaje ensamblador permite al usuario una mayor comprensión del funcionamiento del programa debido a que cada instrucción de programa se corresponde con una instrucción del microcontrolador, por otra banda, al ir aumentando en dificultad, éste programa hace más pesada la programación y aumenta el tiempo de realización, por ello se utiliza un compilador en C en las últimas sesiones para aligerar un poco el trabajo, aunque sería igualmente factible su implementación en MPLAB. El compilador utilizado es el llamado PIC C compiler, de la casa CCS, éste es un compilador muy utilizado por aficionados y profesionales debido a su simplicidad y fácil comprensión.

MPLAB IDE

EL MPLAB IDE es una herramienta software de "Entorno de Desarrollo Integrado" (Integrated Development Enviroment, IDE) que se ejecuta bajo Windows. Con este entorno se pueden desarrollar aplicaciones para los microcontroladores PIC.

El MPLAB incluye todas las utilidades necesarias para la realización de proyectos con microcontroladores PIC, permite editar el archivo fuente del proyecto, además de ensamblarlo y simularlo en pantalla para comprobar cómo evolucionan tanto la memoria de datos RAM, como la de programa ROM, los registros SFR, etc., según progresa la ejecución del programa.

El MPLAB incluye:

Un editor de texto.

Un ensamblador llamado MPASM.

Un simulador llamado MPLAB SIM.

Un organizador de proyectos y otros.

PIC C Compiler (CCS)

El compilador C de CCS ha sido desarrollado específicamente para PIC MCU, obteniendo la máxima optimización del compilador con estos dispositivos. Dispone de una amplia librería de funciones predefinidas, comandos de prerocesado y ejemplos. Además, suministra los controladores (drivers) para diversos dispositivos como LCD, convertidores AD, relojes en tiempo real, EEPROM serie, etc.

Un compilador convierte el lenguaje de alto nivel a instrucciones en código máquina; un cross-compiler, como en el caso de éste compilador, es un compilador que funciona en un procesador (normalmente un PC) diferentes al procesador objeto. Los programas son editados y compilados a instrucciones máquina en el entorno de trabajo del PC, el código máquina puede ser cargado del PC al sistema PIC desde cualquier programador de IC y puede ser depurado desde el entorno de trabajo del PC.

El CCS es C estándar y, además de las directivas estándar (#include, etc.), suministra unas directivas específicas para PIC (#device,etc.); además incluye funciones específicas (bit_set(),etc.). Se suministra con un editor que permite controlar la sintaxis del programa.

4.6. Programas en ensamblador

4.6.1. Introducción programación ensamblador(tutorial MPLAB)

Con ésta primera práctica se pretende introducir al lector en el manejo del programa MPLAB de microchip. Para ello se realizará una breve introducción de las diferentes herramientas del programa y se procederá a la realización de un programa de forma completamente guiada para servir como referencia para prácticas posteriores.

4.6.2. Coche fantástico

Con ésta práctica se pretende trabajar con los diferentes puertos de Entrada/Salida del microcontrolador. Con éste objetivo se utilizarán 6 interruptores conectados al puerto A del microcontrolador y 8 diodos LED conectados al puerto C. De ésta forma se remarcan 3 objetivos distintos en ésta práctica con diferentes niveles de complejidad:

a) Parpadeo de un LED del puerto C independientemente del puerto A.

b) Encender los LED en función de los interruptores (Puerto A = puerto C)

c) Coche fantástico: Se debe profundizar un poco más en el objetivo anterior y conseguir que en función del bit 1 del puerto A los LED del puerto C vayan rotando hacia la izquierda o hacia la derecha.

Para realizar el parpadeo de un Led se debe encender éste y luego apagarlo, cosa que parece muy sencilla, pero cabe tener en cuenta que se debe realizar un retraso entre éstas dos acciones puesto que en caso contrario no sería posible visualizar el cambio. Para realizar éste retraso se procede a crear una rutina de espera, la cual es definida al final del programa (Retardo) y es llamada desde éste (CALL Retardo). Para crear este retraso lo único que se intenta es hacer perder el tiempo al microcontrolador haciendo restas consecutivas. El tiempo de retraso puede ajustarse variando el valor de los números a restar, que deben ser definidos al principio del programa.

Programa en ensamblador para el parpadeo de 1 Led:

```
      __CONFIG  _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF
; Configuración para el programador


      LIST p=16F877A
      INCLUDE <P16F877A.INC>
```

```
N               EQU 0x00
cont1 EQU 0x20    ;Variables para
cont2 EQU 0x21    ;rutina retardo

        ORG 0x00 ; Inicio de programa

Inicio
        BCF STATUS,RP0   ; Accede a
        BCF STATUS,RP1   ;banco 0
        CLRF PORTC ; Limpia PORTC
        BSF STATUS,RP0 ; Accede a banco 1
        CLRF TRISC ; Configura todos las patitas de PORTC como salidas
        BCF STATUS,RP0 ; Regresa a banco 0
Led
        BSF PORTC,0 ; La línea RC0 de PORTC toma el valor de 1, se enciende el LED
        CALL Retardo ; Llamada a la rutina de retardo
        BCF PORTC,0 ; La línea RC0 de PORTC toma el valor de 0, se apaga el LED
        CALL Retardo ; Llamada a la rutina de retardo
        GOTO Led ; Va a la etiqueta Led

Retardo ;Rutina de retardo
        MOVLW N
        MOVWF cont1
Rep1
        MOVLW N
        MOVWF cont2
Rep2
        DECFSZ cont2,1
        GOTO Rep2
        DECFSZ cont1,1
        GOTO Rep1
        RETURN ;Retorno a la llamada de rutina de retardo.
```

```
END ;Fin de programa
```

A continuación se muestran dos programas que nos permiten la rotación del puerto C, el primero se basa en la acción (BSF) y sería más ineficiente debido a que necesitas mas código. El segundo, se basa en la acción (RLF) que nos permite la rotación automática del puerto lo que reduce la cantidad de código por lo que lo hace más eficiente.

Programa para la rotación puerto C (Ineficiente)

```
LIST p=16F877A
INCLUDE <P16F877A.INC>

ORG 0x00 ; Inicio de programa

N EQU 0x00
cont1 EQU 0x20
cont2 EQU 0x21

BCF STATUS,RP0 ; Accede a banco 0
BCF STATUS,RP1
CLRF PORTC ; Limpia PORTC
BSF STATUS,RP0 ; Accede a banco 1
CLRF TRISC ; Configura todos las patitas de PORTC como salidas
BCF STATUS,RP0 ; Regresa a banco 0

LedSec

      BSF PORTC,0 ; La línea RC0 de PORTC toma el valor de 1, se enciende el LED
      CALL Retardo ; Llamada a la rutina de retardo
      BSF PORTC,1 ; La línea RC1 de PORTC toma el valor de 1, se enciende el LED
      CALL Retardo ; Llamada a la rutina de retardo
      BSF PORTC,2 ; La línea RC2 de PORTC toma el valor de 1, se enciende el LED
      CALL Retardo ; Llamada a la rutina de retardo
      BSF PORTC,3 ; La línea RC3 de PORTC toma el valor de 1, se
```

```
enciende el LED
        CALL Retardo ; Llamada a la rutina de retardo
        BSF PORTC,4 ; La línea RC4 de PORTC toma el valor de 1, se enciende el LED
        CALL Retardo ; Llamada a la rutina de retardo
        BSF PORTC,5 ; La línea RC5 de PORTC toma el valor de 1, se enciende el LED
        CALL Retardo ; Llamada a la rutina de retardo
        BSF PORTC,6 ; La línea RC6 de PORTC toma el valor de 1, se enciende el LED
        CALL Retardo ; Llamada a la rutina de retardo
        BSF PORTC,7 ; La línea RC7 de PORTC toma el valor de 1, se enciende el LED
        CALL Retardo ; Llamada a la rutina de retardo


        GOTO LedSec ; Va a la etiqueta LedSec


Retardo ;Rutina de retardo
        MOVLW N
        MOVWF cont1
Rep1
        MOVLW N
        MOVWF cont2
Rep2
        DECFSZ cont2,1
        GOTO Rep2
        DECFSZ cont1,1
        GOTO Rep1
        RETURN ;Retorno a la llamada de rutina de retardo.


END ;Fin de programa
```

Programa para la rotación puerto C (Eficiente)

```
__CONFIG _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF
; Configuración para el programador

LIST p=16F877A
INCLUDE <P16F877A.INC>

ORG 0x00 ; Inicio de programa

N EQU 0x00
cont1 EQU 0x20
cont2 EQU 0x21

BCF STATUS,RP0 ; Accede a banco 0
BCF STATUS,RP1
CLRF PORTC ; Limpia PORTC
BSF STATUS,RP0 ; Accede a banco 1
CLRF TRISBC ; Configura todas las patitas de PORTC como salidas
BCF STATUS,RP0 ; Regresa a banco 0

BSF PORTC,0 ; La línea RC0 toma el valor 1, se enciende el LED
Ledsec
CALL Retardo ; Llamada a la rutina de retardo
RLF PORTC,1 ; Recorre el bit de RC0 a RC7
GOTO Ledsec ; Va a la etiqueta Ledsec

Retardo ; Rutina de retardo
MOVLW N
MOVWF cont1
Rep1
MOVLW N
MOVWF cont2
```

```
Rep2
DECFSZ cont2,1
GOTO Rep2
DECFSZ cont1,1
GOTO Rep1
RETURN ; Retorno a la llamada de rutina de retardo.

END ; Fin de programa
```

4.6.3. Visualización dinámica

En la presente práctica se pretende realizar un programa en lenguaje ensamblador para implementar un contador decimal de 0000 a 9999.

Para la correcta visualización de los valores se deberá utilizar una rutina de retardo para visualizar el valor unas pocas veces antes de incrementarlo.

Los displays ya se encuentran conectados a un driver BCD/7 segmentos externo para el ahorro de pines del microcontrolador. Este driver está conectado a la parte baja del puerto, y los cuatro bits altos son los que controlan los transistores que habilitan los displays.

Para familiarizarse con el control de displays se ha realizado un programa que actúa sobre un único display creando un contador desde 0 hasta 9. Para ello envía sucesivamente el código de cada uno de ellos. Éste es un programa bastante ineficiente pero que puede ayudar a observar el funcionamiento del codificador externo así como el correcto funcionamiento de los distintos displays. Posteriormente se intentará reducir el código del programa y se utilizará la visualización dinámica, es decir, se actuará sobre los transistores, para hacer el contador desde 0 hasta 9999.

Programa para implementar un contador 0-9 1 display

```
    __CONFIG  _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF
; Configuración para el programador

    LIST p=16F877A
    INCLUDE <P16F877A.INC>

    ORG 0x00 ; Inicio de programa
```

```
N EQU 0x00
cont1 EQU 0x20
cont2 EQU 0x21

        BCF STATUS,RP0 ; Accede a banco 0
        BCF STATUS,RP1
        CLRF PORTC ; Limpia PORTC
        CLRF PORTD ; Limpia PORTC
        BSF STATUS,RP0 ; Accede a banco 1
        CLRF TRISC ; Configura todos las patitas de PORTC como salidas
        CLRF TRISD ; Configura todos las patitas de PORTD como salidas
        BCF STATUS,RP0 ; Regresa a banco 0

Display

        movlw       b'10000000'       ;1000(control)0000(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo

        movlw       b'10000001'       ;1000(control)0001(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo

        movlw       b'10000010'       ;1000(control)0010(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo

        movlw       b'10000011'       ;1000(control)0011(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo

        movlw       b'10000100'       ;1000(control)0100(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo
```

```
        movlw       b'10000101'         ;1000(control)0101(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo

        movlw       b'10000110'         ;1000(control)0110(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo

        movlw       b'10000111'         ;1000(control)0111(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo

        movlw       b'10001000'         ;1000(control)1000(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo

        movlw       b'10001001'         ;1000(control)1001(datos)
        movwf       PORTD
        CALL Retardo ; Llamada a la rutina de retardo



        GOTO Display ; Va a la etiqueta Display

Retardo ;Rutina de retardo
        MOVLW N
        MOVWF cont1
Rep1
        MOVLW N
        MOVWF cont2
Rep2
        DECFSZ cont2,1
        GOTO Rep2
        DECFSZ cont1,1
        GOTO Rep1
        RETURN ;Retorno a la llamada de rutina de retardo.
```

```
END ;Fin de programa
```

4.6.4. Contador LCD

En ésta práctica se pretende seguir el mismo esquema que en la práctica anterior, pero en este caso para realizar un contador en un display de cristal líquido (LCD).

4.6.5. Llave electrónica

En ésta práctica se pretende introducir al lector en la utilización de un teclado matricial. Para ello se realizará un programa que guarde un código secreto en el microcontrolador, al introducir éste código desde el teclado se deberá encender un led para indicar que el valor es el correcto.

4.6.6. Sensado temperatura

En ésta práctica se pretende estudiar el funcionamiento del convertidor analógico digital incorporado en el microprocesador. Para ello nos ayudaremos del sensor de temperatura conectado al microcontrolador y de los conocimientos adquiridos previamente para la visualización (LCD o 7 segmentos) para adquirir la señal proporcionada por el sensor e indicar la temperatura ambiente.

4.6.7. Control con PWM

En ésta práctica se pretende trabajar con el controlador PWM incorporado en el microcontrolador. Para ello se plantean dos opciones posibles:

- Control iluminación Led
- Control sonido (Variante de PWM en que se varía también la frecuencia)

4.7. Programas en C

4.7.1. Introducción programación C (tutorial PIC C)

De la misma forma que se procedió en la primera práctica con ensamblador, Con ésta práctica se pretende introducir al lector en el manejo del programa compilador PIC C. Para ello se realizará una breve introducción de las diferentes herramientas del programa y se procederá a la realización de un programa de forma completamente guiada para servir como referencia para prácticas posteriores.

4.7.2. Sonidos con PWM

En ésta práctica se pretende complementar el trabajo de la última práctica realizada en ensamblador. Para ello se realizará el mismo programa pero en lenguaje C, para observar así las prestaciones de éste.

Es programa siguiente es un programa que proporciona un control PWM sin utilizar el módulo incorporado en el microcontrolador, para ello utiliza variables que serán los tiempos en estado alto y bajo.

Control Sonido por teclado

```
#include <16f877A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#USE DELAY (CLOCK=4000000)
#INCLUDE <kbd.c>
#USE STANDARD_IO (a)

VOID MAIN()
{
  CHAR k,kant='0';     ;k valor de teclado, k valor anterior de teclado
  char PWMH=0,PWML=0;  ;Semiperiodo alto y bajo
  kbd_init();
  PORT_B_PULLUPS(TRUE);

WHILE (1) {
    k=kbd_getc();          ;Lee el valor de la tecla pulsada
    if (k=='\0') k=kant;       ;Si no se pulsa tecla (\0) se usa el valor
anterior
```

```
    if ((k=='*') || (k=='#')) k='0'; ;Si se pulsa * o # se asigna un valor cero.
    kant=k;                    ;Se guarda tecla pulsada
     k=k-48;                    ;Se convierte de ASCII a valor numérico
     PWMH=k*28;                  ;Proporción entre valor tecla y semiperiodo Alto.
     PWML=255-PWMH;              ;Semiperiodo Bajo
    for(PWMH;PWMH>0;PWMH--){       ;Obtención de la salida a nivel alto
     OUTPUT_HIGH(PIN_A0);}
    for(PWML;PWML>0;PWML--){     ;Obtención de la salida a nivel bajo
    OUTPUT_LOW(PIN_A0);}
 }
```

4.7.3. Control reloj I2C

En ésta práctica se estudia el bus de comunicaciones I2C así como el reloj en tiempo real DS1307. Para ello se propone realizar un programa que adquiera la hora del reloj y la visualice por el LCD.

4.7.4. Comunicación puerto serie

A continuación se muestra un programa sencillo para el PIC16F877A que se encarga de enviar un dato adquirido por el microcontrolador a través del puerto serie, en éste casos e envía tanto el valor de voltaje como el del convertidor. Además, se incluyen unas imágenes con los pasos a seguir para configurar el hyperterminal de Windows para crear una conexión nueva y poder comunicar el microcontrolador con el PC.

Programa comunicación Puerto Serie

```
#include <16F877A.h>
#device adc=10
#FUSES XT,NOWDT
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=pin_c6, rcv=pin_c7, bits=8, parity=N)
#include <LCD.C>

void main() {
```

```
  int16 q;
  float p;
  setup_adc_ports(AN0);
  setup_adc(ADC_CLOCK_INTERNAL);
  lcd_init();
  for (;;) {
    set_adc_channel(0);
    delay_us(10);
    q = read_adc();
    p = 5.0 * q / 1024.0;
    printf(lcd_putc, "\fADC = %4ld", q);
    printf(lcd_putc, "\nVoltage = %01.2fV", p);
    printf("ADC = %4ld ", q);
    printf("Voltage = %01.2fV\r", p);
    delay_ms(100);

    }
}
```

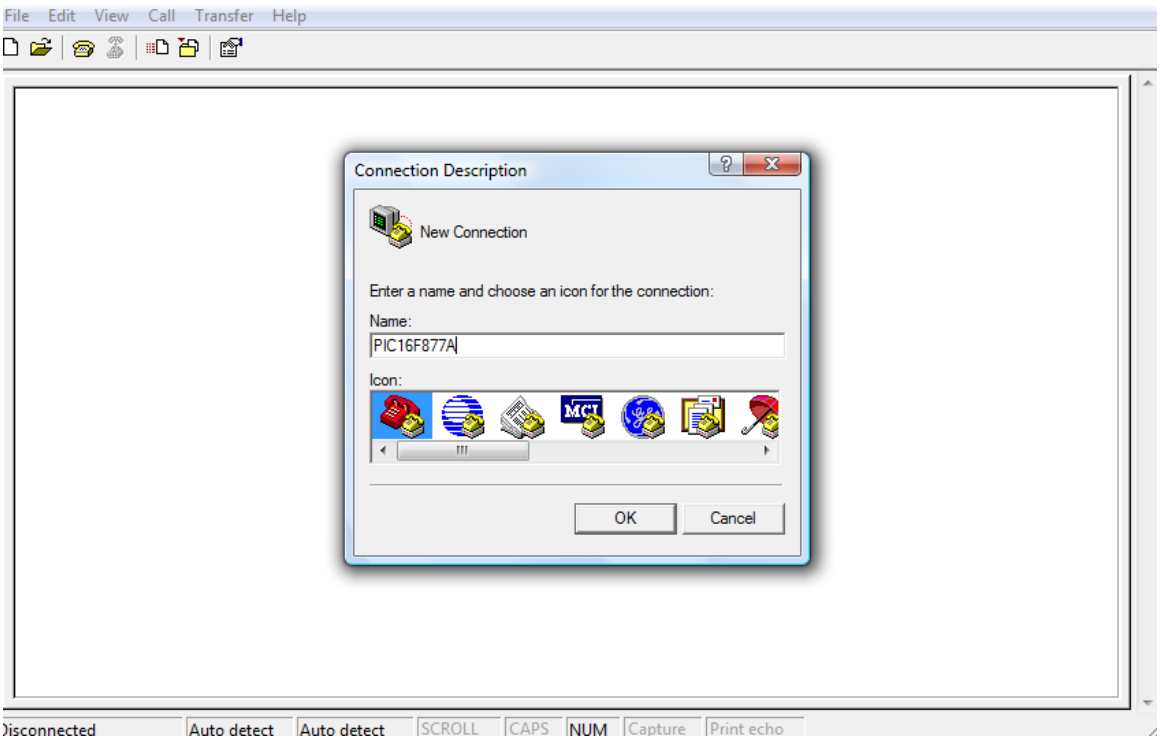


Figura 17. *Abrimos una nueva conexión en el hyperterminal*

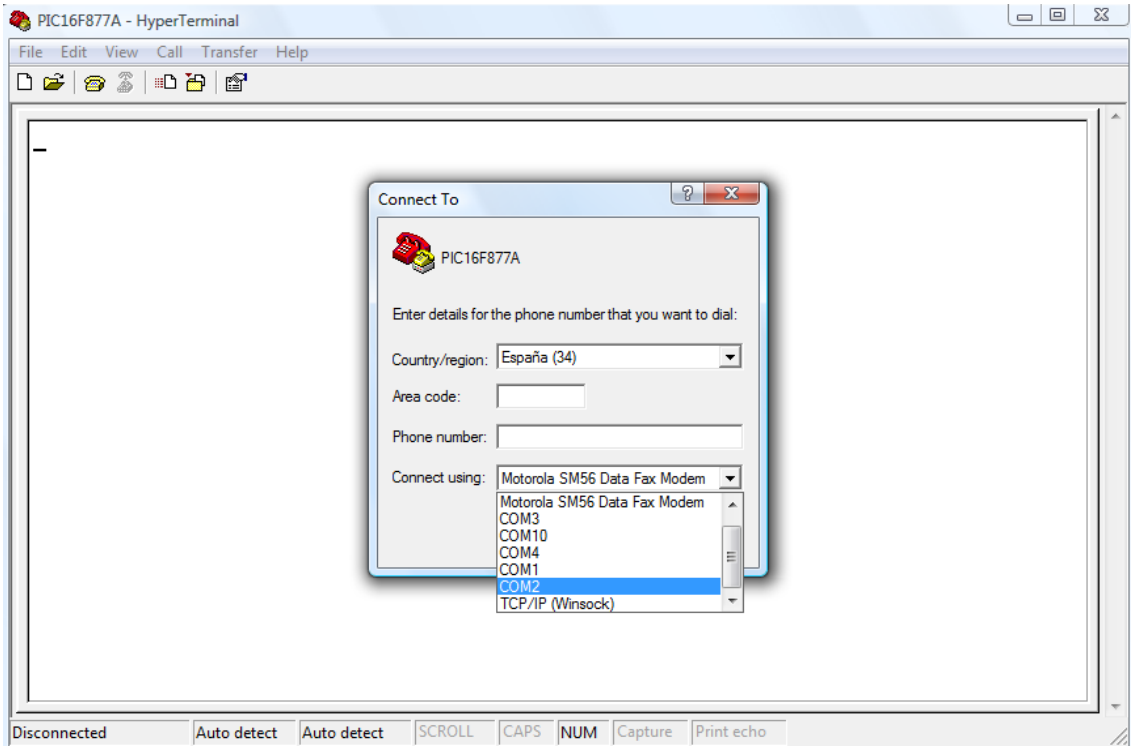


Figura 18. *Elegimos el puerto de comunicaciones*

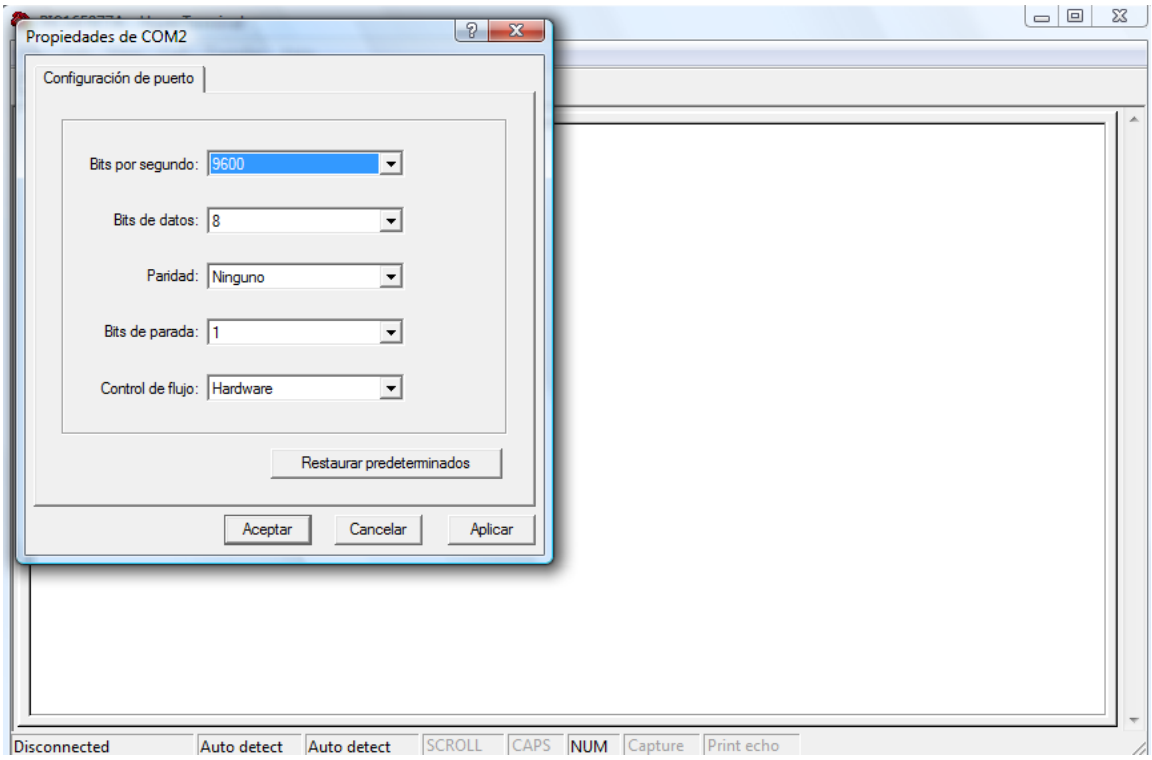


Figura 19. *Configuramos el puerto de la misma forma que el microcontrolador*

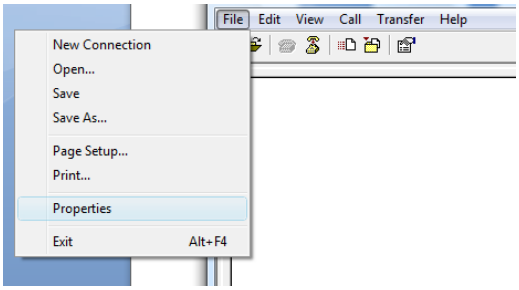


Figura 20. *Agregamos avance de línea (1)*

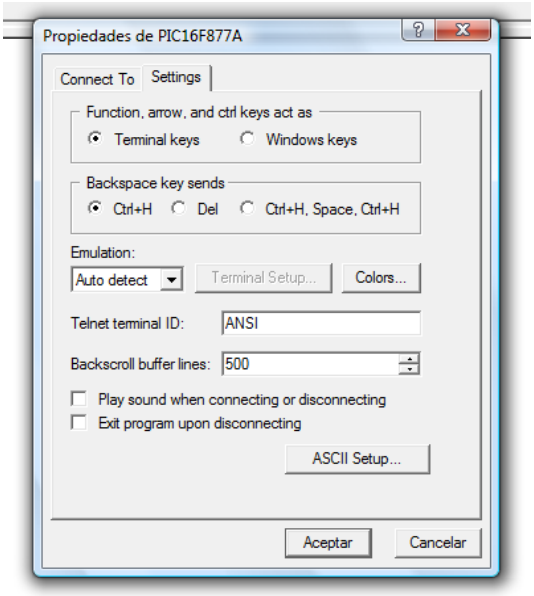


Figura 21. *Agregamos avance de línea (2)*

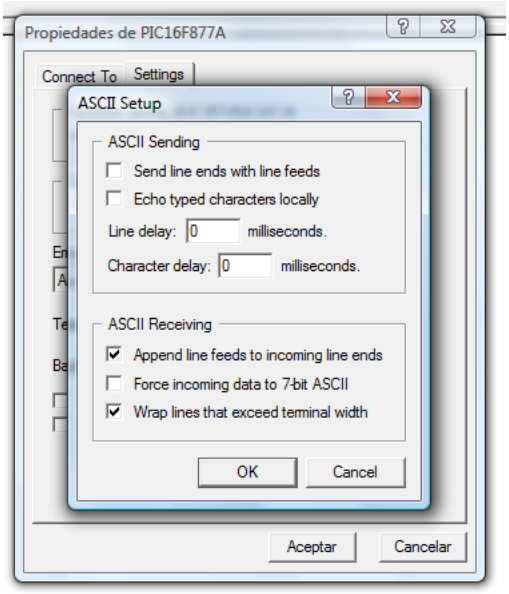


Figura 22*. Agregamos avance de línea (3)*

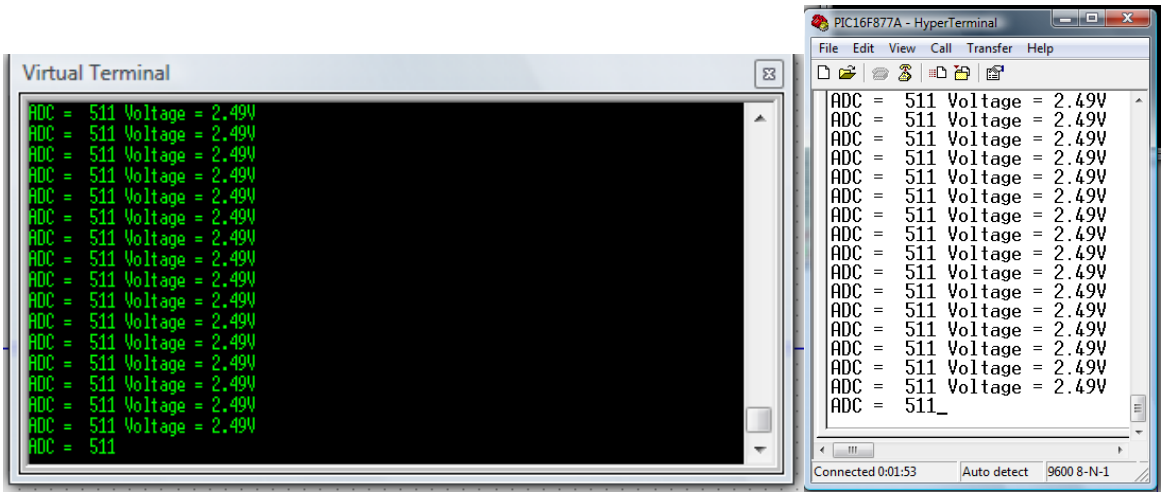


***Figura 23**. Circuito funcionando en proteus*

4.7.5. Control supervisado labview

En la siguiente práctica se pretende utilizar todos los conocimientos conseguidos con prácticas anteriores para crear un sistema de control supervisado. El tema es libre, pero debería consistir, como mínimo, en visualizar una señal medida por el microcontrolador, de la misma forma que la práctica anterior pero realizando el programa en labview, además ésta deberá ser graficada.

El siguiente programa toma una medida del microcontrolador y la gráfica en labview, además extrae los valores máximos y mínimos, crea una tabla de valores, etc... Una de las características principales es que se ha utilizado el control de flujo por hardware, un método prácticamente desconocido, pero muy útil en aplicaciones de éste tipo y que viene bien conocer.

Éste programa se encuentra actualmente en desarrollo, debido a que el trabajo con labview no tiene límite, y uno puede diseñar prácticamente todo aquello que se proponga. Actualmente se hasta trabajando en una comunicación bidireccional para permitir al PC actuar sobre el microcontrolador y así crear un sistema en lazo cerrado.

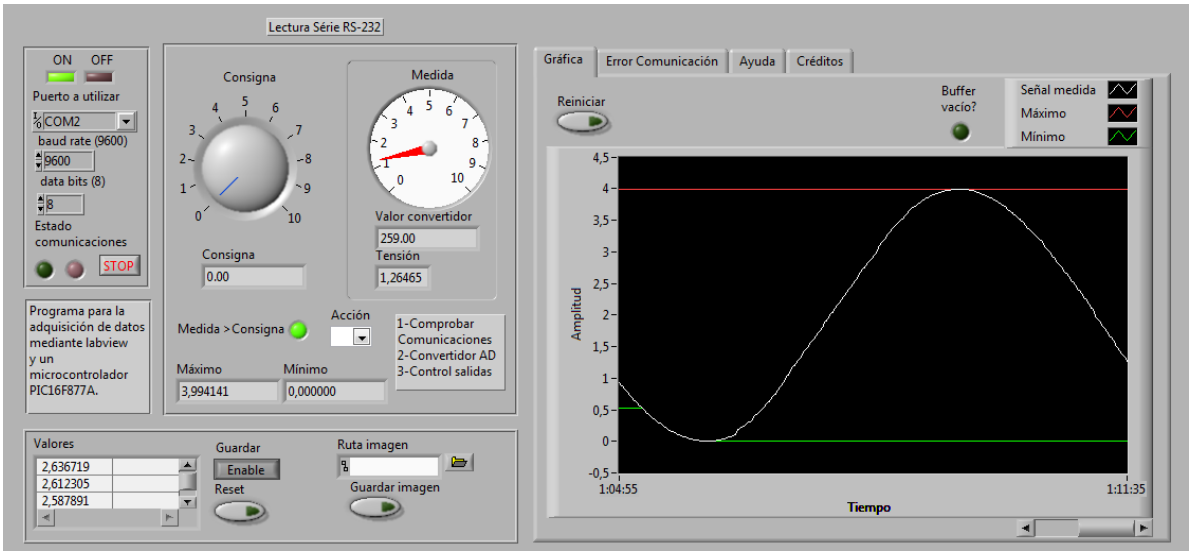


Figura 24. *Panel frontal en funcionamiento*

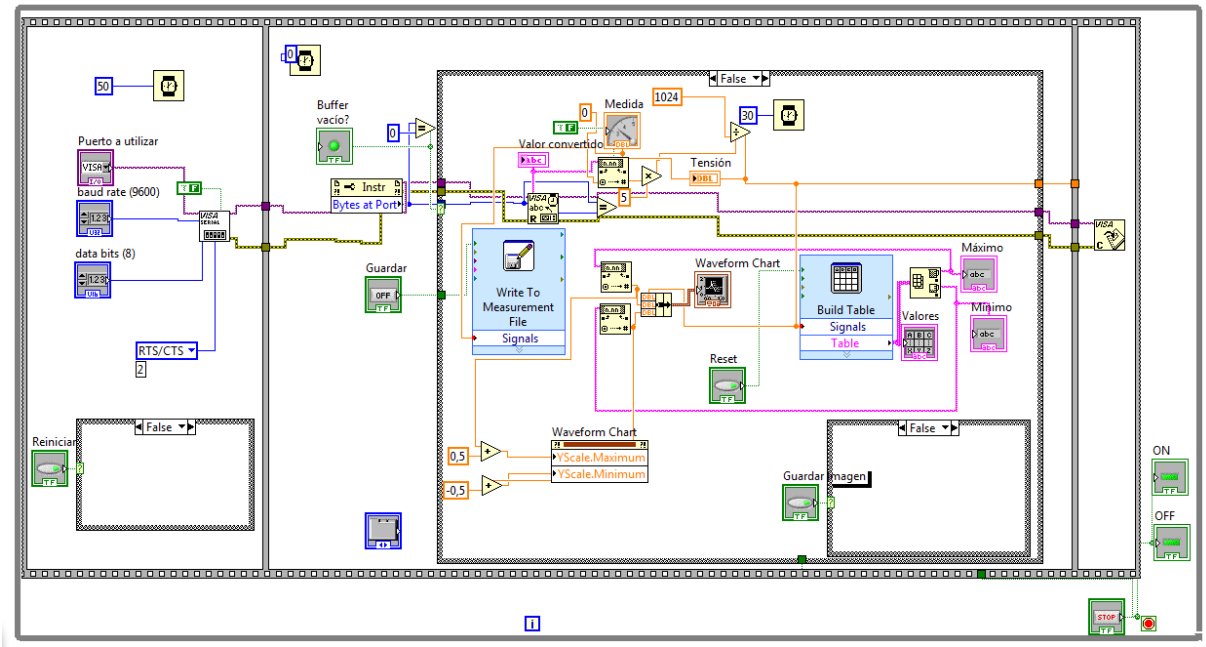


Figura 25. *Código programa labview*

Programa control supervisado

```
#include <16F877A.h>
#device adc=10
#FUSES XT,NOWDT
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=pin_c6, rcv=pin_c7, bits=8, parity=N)
#include <LCD.C>
#BYTE TRISC =0X87
#BYTE PORTC =0X07
```

```
float ch;
float p,q;

//#int_rda
//void_serial(){


//   bit_clear(PORTC,2);
//   delay_ms(20);
//}


void main() {

  int flag=0;

  bit_set(TRISC,1);
  bit_clear(TRISC,0);
  bit_clear(TRISC,2);

  setup_adc_ports(AN0);
  setup_adc(ADC_CLOCK_INTERNAL);
  lcd_init();


  for (;;) {
    set_adc_channel(0);
    delay_us(10);
    q = read_adc();

    bit_set(PORTC,0);

    if((bit_test(PORTC,1)==1)){
```

```
        bit_clear(PORTC,0);
        delay_ms(20);
        printf("%01.2f",q);  // El \r permite cambiar de línea.
        while((bit_test(PORTC,1)==1)){
          bit_set(PORTC,0);

        }
    }

    }
}
```

CAPÍTULO 6: BIBLIOGRAFIA

5.1. Bibliografía de Consulta

Enrique Palacios, Fernando Remiro, Lucas J. López. Microcontrolador PIC16F84A Desarrollo de proyectos (3ª Edición). Ediciones Ra-Ma 2009.

Germán Tojeiro Calaza. PROTEUS, Simulación de circuitos electrónicos y microcontroladores a través de ejemplos. Ediciones Marcombo, Enero 2009.

Eduardo García Breijo. Compilador C CCS y simulador Proteus para microcontroladores PIC (2ª Edición). Ediciones Marcombo, 2009.

www.microchip.com

http://www.micropic.es/

http://www.ucontrol.com.ar

www.todopic.es

Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Anexo II.
Manual de Usuario

"DISEÑO DE UNA PLATAFORMA DOCENTE PARA EL APRENDIZAJE DE MICROCONTROLADORES 'PIC' DE MICROCHIP"

PFC presentado para optar al título de Ingeniero Técnico Industrial especialidad ELECTRÓNICA INDUSTRIAL

por **Víctor Bueno Álvez**

Barcelona, 12 de Enero de 2011

Tutor proyecto: Herminio Martínez García
Departamento de Ingeniería electrónica (DEEL)
Universitat Politècnica de Catalunya (UPC)

ÍNDICE ANEXO II: MANUAL DE USUARIO

II.1 Manual de usuario

II.1.1 Introducción

Con este documento se pretende introducir al usuario de una forma más concreta, a las características y funcionamiento de la entrenadora de microcontroladores PIC diseñada en este proyecto. Todos los fragmentos redactados en el manual de usuario se pueden encontrar representados íntegramente en la memoria técnica del proyecto, y se encuentran representados aquí para facilitar la tarea de lectura para el usuario final. A este manual de usuario se ha añadido un manual de componentes, en donde se definen de manera breve las características principales de los diferentes periféricos utilizados en el circuito, con el objetivo de servir de ayuda para usuarios iniciados en el tema.

II.1.2 Características

La placa entrenadora PIC-vBoard es un conjunto didáctico en donde podemos trabajar con los microcontroladores PIC de la marca microchip.

Esta entrenadora está diseñada para trabajar con un PIC16F877A lo que nos permite trabajar con los PIC de 40 pines más importantes de la serie 16F y 18F, así como el resto de PICs de las series 12F, 16F y 18F que no sean compatibles con éste mediante un zócalo adaptador. En la placa se incluyen una gran variedad de periféricos para probar sobre ésta nuestros circuitos con microcontroladores tanto a nivel software como hardware.

Todos estos periféricos se encuentran directamente conectados con las diferentes patillas del microcontrolador, por lo que el usuario no deberá hacer ningún tipo de conexión, lo que hace aún más fiable su funcionamiento y permitirá centrarse únicamente en los errores de software.

Las principales características de la placa son:

- Es un circuito fácilmente expandible gracias a un conector DIL-20 que recoge varios puertos, señales de entrada y salida analógicas y las señales de alimentación.
- Dispone de una alimentación a 230V lo que permite conectarla directamente a la red sin necesidad de ninguna fuente de alimentación. Se integra en el mismo circuito una alimentación doble de 5V para alimentar independientemente, el microcontrolador y los periféricos de bajo consumo de los de alto consumo y la alimentación exterior. Cada alimentación incluye una etapa transformadora, rectificadora, de filtrado y estabilización así como un LED indicador de encendido.
- Se admiten microcontroladores PIC de 40 patillas compatibles con PIC16F877A, así como otros microcontroladores no compatibles utilizando un zócalo adaptador.
- Contiene un oscilador de cuarzo de 4MHZ para generar la frecuencia de trabajo del microcontrolador, aunque éste puede ser cambiado fácilmente por otro del valor deseado.
- Todos los puertos del PIC son accesibles ya sea mediante el conector DIL-20 o mediante los conectores SIL de algunos componentes después de haber extraído éstos, como es el caso del puerto D en el LCD o el C en la Barra de LEDS.
- Comunicación RS232 integrada con conector DB9 hembra y control de las señales TxD, RxD, CTS y RTS mediante el integrado MAX232.
- Comunicación I2C con Reloj Calendario en tiempo Real y convertidor Analógico Digital y Digital Analógico.
- 8 salidas digitales mediante una barra de Leds verdes.
- 4 displays de 7 segmentos de 0.52 y ánodo común controlados por un decodificador BCD/7 segmentos que permite practicar las técnicas de visualización dinámica.
- Pantalla LCD de 2x16 con comunicación de 4 bits y jumper externo para activación de retro iluminación.
- Teclado matricial de 16 teclas con teclas alfanuméricas y símbolos.
- Entradas digitales formadas por 3 pulsadores y 3 interruptores.
- 4 entradas analógicas seleccionables en lugar de las digitales formadas por 2 potenciómetros con cursor manejables manualmente, un sensor de temperatura y uno de luminosidad.
- Conector DIL-20 para acceder a las principales señales del circuito desde el exterior.

Especificaciones técnicas

Una vez se han decidido las características que debe cumplir nuestro circuito se procede a diseñar una lista con las especificaciones técnicas de la entrenadora, que como se ha comentado anteriormente, sería la siguiente:

- Fuente de alimentación doble integrada de +5/+5 V de 800 mA cada una con etapa transformadora, rectificador, y de filtrado y estabilización.
- 8 salidas digitales formadas por una barra de 10 LEDs verdes.
- 6 entradas digitales: 3 mediante pulsadores y 3 con interruptores.
- 4 entradas analógicas conmutables una a una con las entradas analógicas, formadas por:
    a. 2 potenciómetros con cursor.
    b. 1 sensor de temperatura Lm35.
    c. 1 sensor de luminosidad LDR.
- 1 teclado matricial 4x4.
- 1 visualizador LCD 2x16.
- 4 displays 7 segmentos en color rojo y ánodo común con transistores de conmutación y controlador 4543 para decodificación BCD/7seg.
- 1 Reloj/calendario en tiempo real (RTC) en bus I2C modelo DS1307.
- 1 convertidor AD/DA en bus I2C modelo PCF8591 con 4 entradas y una salida analógica.
- Comunicaciones: MAX232 con conector DB9 y USB.
- Ampliación: Conector DIL-20.

II.1.3 Arquitectura

En este apartado se muestra el diagrama de bloques de la tarjeta para que el usuario pueda hacerse una idea de los periféricos incluidos de una forma más general. Se pueden observar a que puerto están conectados los diferentes periféricos así como aquellos que comparten un mismo puerto.

Algunos elementos, como los conectores de ampliación, no están incluidos para simplificar el diagrama y centrar la atención en los periféricos más importantes del circuito.

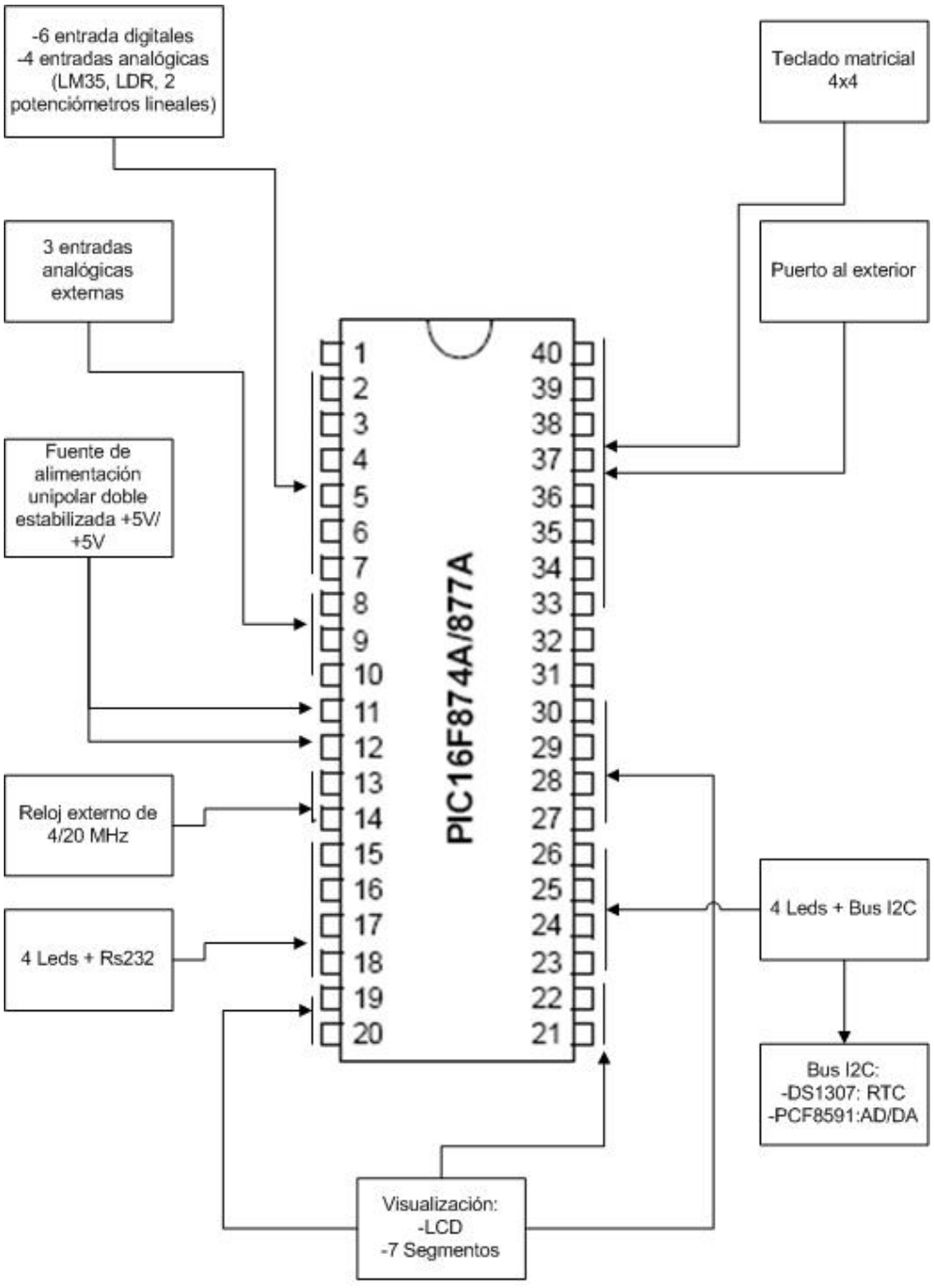


Figura 1. *Diagrama de bloques*

II.1.4 Definición bloques

Una vez presentado el diagrama general de la aplicación, se procederá a introducir brevemente los diferentes bloques, mostrando sus características y su función en la tarjeta, además de los cálculos realizados en caso de que fuera oportuno.

Fuente de alimentación

La fuente de alimentación está formada por un transformador de 12 VA y 230 V de entrada, protegidos por un fusible de 1,25 A con un interruptor en serie para encender y parar el circuito y una salida doble independiente de 7.5 V y 800 mA.

Este transformador es el más pequeño que se ha encontrado en el mercado capaz de proporcionar una tensión suficiente para nuestro integrado 7805. El fabricante de este integrado nos indica que la tensión mínima de entrada para asegurar una tensión estabilizada de 5 V a la salida debe ser de 7 V, por lo tanto considerando una caída de tensión de 1,2 V en los diodos del rectificador de onda completa, necesitamos una tensión de pico de como mínimo 8,2 V a la salida del transformador, con lo cual, la tensión de algo más de 10 V de pico que nos ofrece este transformador es ideal para esta aplicación.

Cada salida posee un rectificador de onda completa encapsulado para la rectificación de la onda, así como un condensador electrolítico de 1000 uF para su estabilización y un integrado LM7805 con un condensador electrolítico de 100 uF y uno cerámico de 100 uF para conseguir una tensión estabilizada de 5V positivos. A este integrado se le ha añadido un disipador para impedir así su sobrecalentamiento. Cabe considerar que las fuentes de alimentación no son totalmente independientes, sino que sus masas están unidas para evitar así posibles problemas en los que una alimentación se derive a la masa de la otra (como en el caso de que se activara un relé desde el PIC). Además, se ha añadido un LED rojo de 5mm a una salida, el cúal, suponiendo una caída de tensión en el led de 1,6V y considerando la tensión de alimentación de 5V, se ha puesto en serie con una resistencia de 330 Ω para conseguir una intensidad media de unos 10 mA, y un led verde a la otra salida con una resistencia de 180 Ω al suponer una caída de tensión de 2,4 V, consiguiendo una intensidad inferior a 20 mA, de esta forma, los leds se enciendan en el momento que haya tensión.

De esta forma diseñamos dos circuitos de alimentación independiente de 5V cada uno, que nos permiten alimentar por una banda el microcontrolador y los periféricos de bajo consumo, y por otra los elementos de mayor consumo y la alimentación externa. Con este método es posible reducir las caídas de tensión producidas en la alimentación a la hora de utilizar periféricos de alto consumo y rápidas conmutaciones, como sería el caso de una visualización dinámica con displays de 7 segmentos.

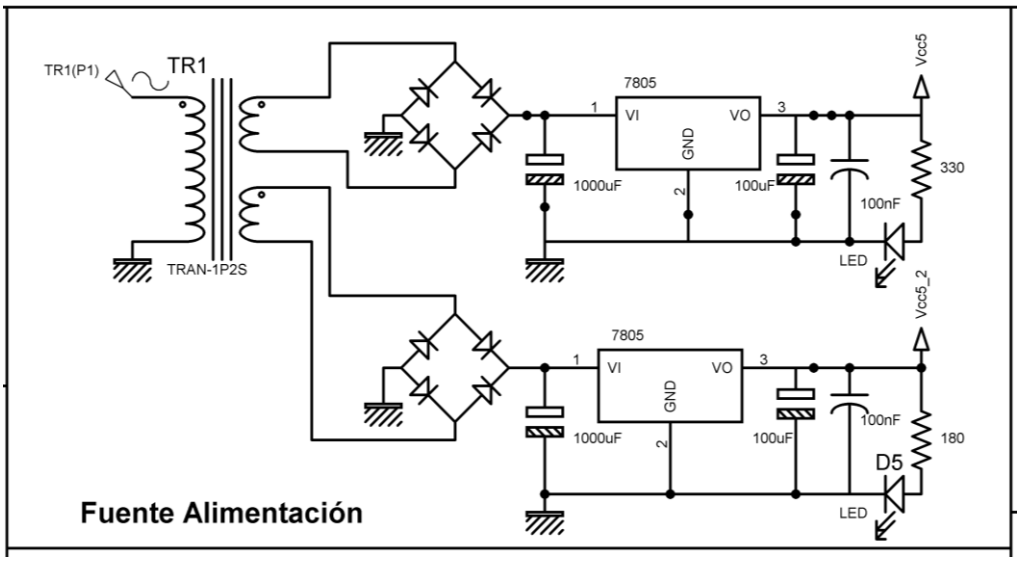


Figura 2. *Fuente de alimentación*

Circuito microcontrolador

Como bloque principal del circuito se encuentra el microcontrolador junto con los elementos que hacen posible su correcto funcionamiento, estos son:

- Circuito oscilador:
  Está compuesto por un cristal de cuarzo de 4 MHz intercambiable, junto con dos condensadores de 22 pF conectados a masa. El valor de estos condensadores para los diferentes microcontroladores y cristales se puede encontrar en la hoja de características del microcontrolador. Este cristal se conecta a las patillas XTAL1 y XTAL2 del microcontrolador y proporciona la frecuencia de trabajo del mismo.
- Circuito de Reset:
  Este circuito nos permite, mediante un pulsador, reinicializar el microcontrolador. Su funcionamiento es simple, el PIC se reinicializa al poner el pin MCLR a masa, por lo tanto se conecta este pin a alimentación mediante una resistencia de 50 KΩ y en paralelo el pulsador con una resistencia de 10 KΩ a masa.

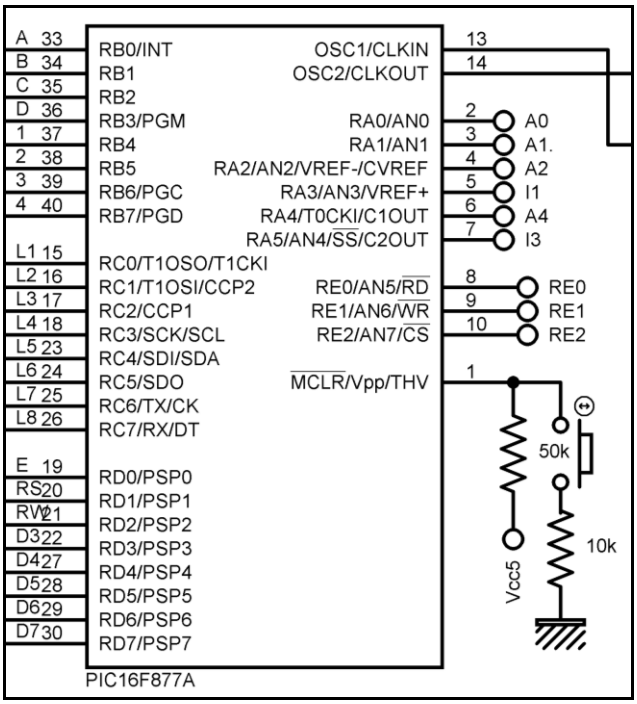


Figura 3. *Circuito microcontrolador*

Salidas Digitales

Las salidas digitales están formadas por una barra de 10 LEDs verdes conectados a una resistencia SIL de 180 Ω con su común conectado a masa de los cuales 8 están conectados al puerto C del microcontrolador y los otros dos se encuentran accesibles para otros usos. Al estar conectados al puerto C cabe considerar que éstos deben ser desconectados a la hora de utilizar las señales PWM, el bus I2C o la comunicación RS232. No es obligatorio en todos los casos pero es muy recomendable hacerlo siempre para evitar posibles errores. Su desconexión se realizará quitando la conexión a masa de las resistencias tipo SIL mediante el jumper correspondiente.

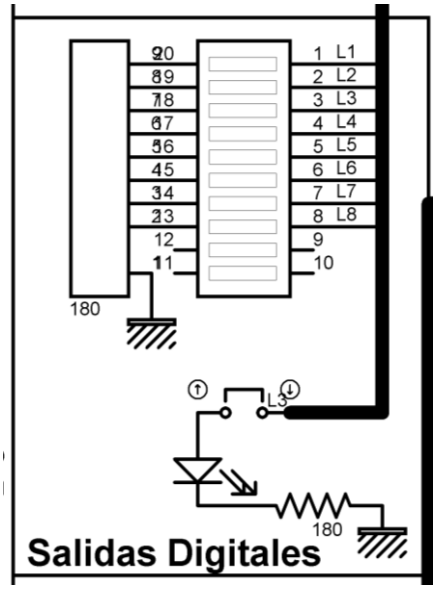


Figura 4. *Salidas digitales*

Salidas a Relé

La placa incluye integradas 2 salidas a relé, un relé clásico y uno de estado sólido. El primero es activado mediante pin RE1 a través de su transistor de activación del tipo 2n2222 y es utilizado para el control simple de dispositivos de corriente alterna de hasta 12 A. El relé de estado sólido es controlado directamente por tensión a través del pin CCP1, lo que nos permite, aprovechando las ventajas de rapidez de conmutación de este tipo de relés, realizar controles PWM sobre dispositivos externos de corriente alterna de hasta 5A.

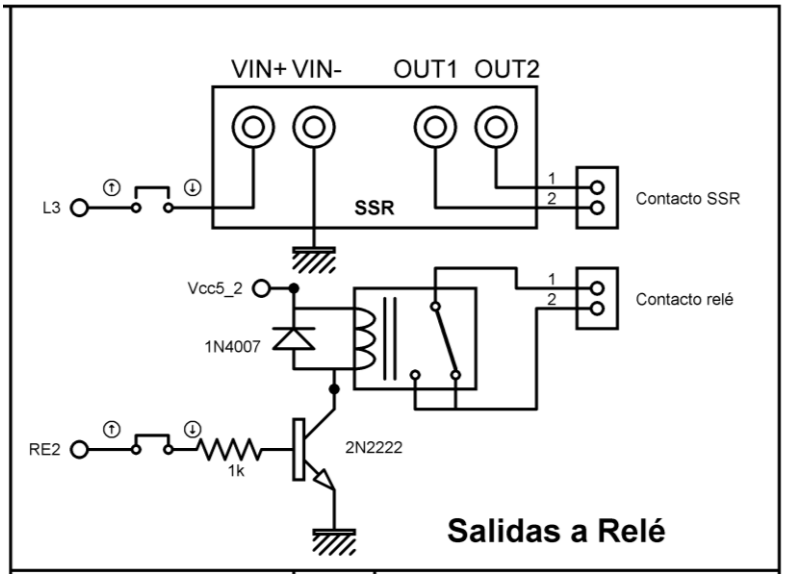


Figura 5. *Salidas a relé*

Salidas digitales extra

Además de las salidas digitales explicadas anteriormente, también se encuentran algunas salidas útiles para la práctica de otros aspectos del microcontrolador.

Por una parte disponemos un led de alta luminosidad con una resistencia de 180 Ω en serie conectado entre masa y el pin CCP1 del microcontrolador, permitiéndonos visualizar sobre la misma placa la variación de iluminación mediante PWM de una forma más clara que con un led corriente.

Por otro lado disponemos una resistencia calefactora en serie con un led indicador conectada al pin RE1 que se encuentra dispuesta enfrente del sensor de temperatura y nos permitirá realizar un control de temperatura en lazo cerrado.

Entradas digitales

Las entradas digitales están formadas por 3 interruptores y 3 pulsadores normalmente abiertos (NO) conectados entre masa y una resistencia tipo SIL de 10 KΩ con su común conectado a 5V. De esta forma las señales se encuentran conectadas a las patillas de cada resistencia, por lo cual las entradas son invertidas, es decir, siempre se encuentran conectadas a alimentación y pasan a conectarse a masa en el caso de activar la entrada.

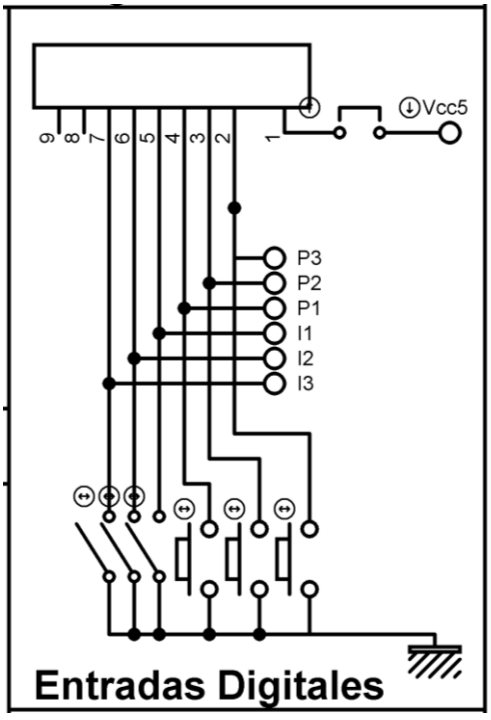


***Figura 6**. Entradas digitales*

Entradas analógicas

Las entradas analógicas se encuentran conmutadas con los tres pulsadores y un interruptor mediante 4 jumpers de tres contactos, lo que permite seleccionar una a una si las entradas serán analógicas o digitales. Además existen dos jumpers para conectar o desconectar la alimentación de las entradas analógicas o digitales, en el caso de que sólo deseemos utilizar unas u otras.

Por un lado tenemos 2 potenciómetros de 10 KΩ conectados entre Vcc y masa y con su punto central conectado a la entrada del microcontrolador para nuestras primeras prácticas con el convertidor AD del microcontrolador.

Además de estas dos entradas tenemos un sensor de temperatura LM35 que dispone de 3 patillas, 2 para la alimentación y la otra es la que se conecta a la entrada del microcontrolador.

Por último disponemos de una LDR, que nos permite medir el nivel de luminosidad, pero ésta no puede ser conectada directamente al microcontrolador sino que se debe realizar un divisor de tensión para poder medir la caída de tensión con el microcontrolador. Esta LDR se encuentra dispuesta cerca del LED de alta luminosidad para poder así realizar un pequeño control en lazo cerrado.

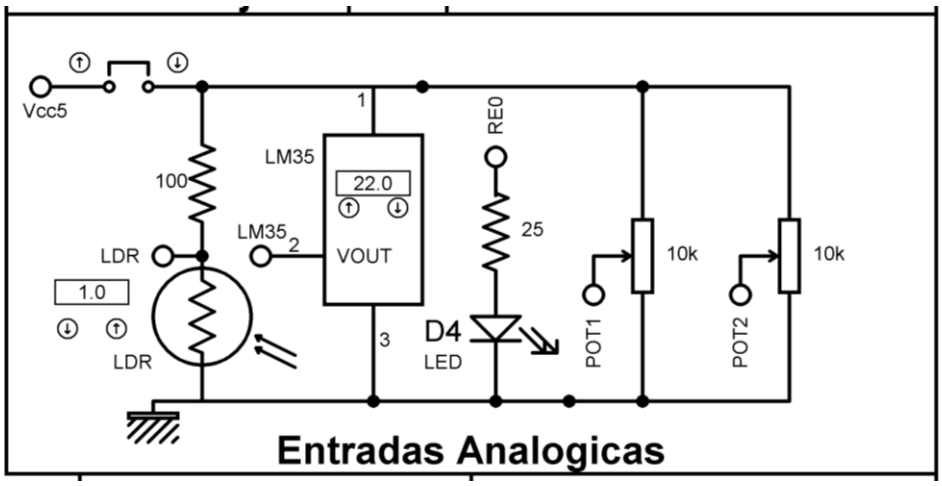


Figura 7. *Entradas analógicas*

Teclado matricial

El teclado matricial se encuentra conectado al puerto B del microcontrolador, aprovechando así las ventajas que tiene este puerto para el control de estos dispositivos como son las resistencias de pull-up y la interrupción por variación de sus pines. Además, cabe tener en cuenta que algunas señales de este puerto son compartidas con las señales de ICSP así que se recomienda no pulsar ninguna tecla durante la programación así como desconectar el programador cuando se esté utilizando el teclado. Además, esté puerto se encuentra disponible al completo a través del conector DIL-20, así que se recomienda extraer el teclado si se utiliza en un circuito externo para evitar daños en caso de pulsar alguna tecla ya que quedarán cortocircuitadas filas y columnas.

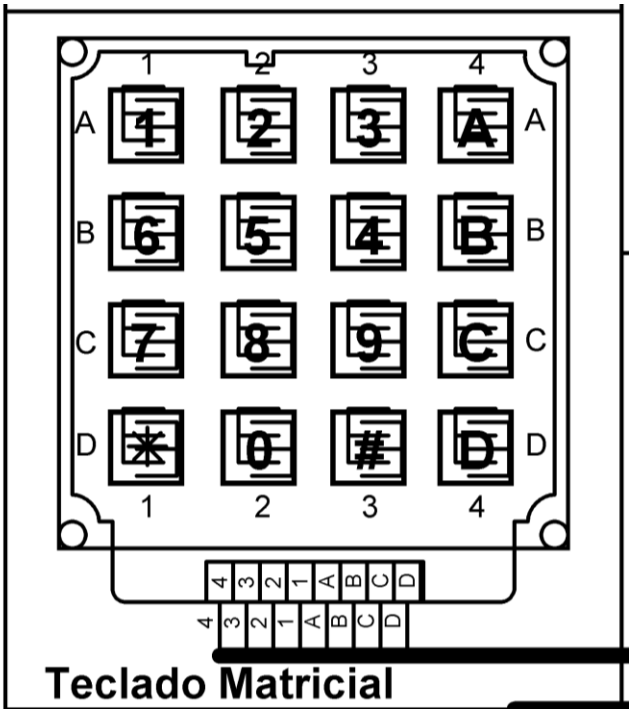


Figura 8. *Teclado matricial*

Visualizador LCD

El display LCD se encuentra conectado al puerto D del microcontrolador y dispone de 1 jumper de 3 contactos para conmutar la alimentación entre el LCD o los displays de 7 segmentos, además tiene una resistencia en serie con el LED de retro iluminación y un jumper para activar o desactivarla.

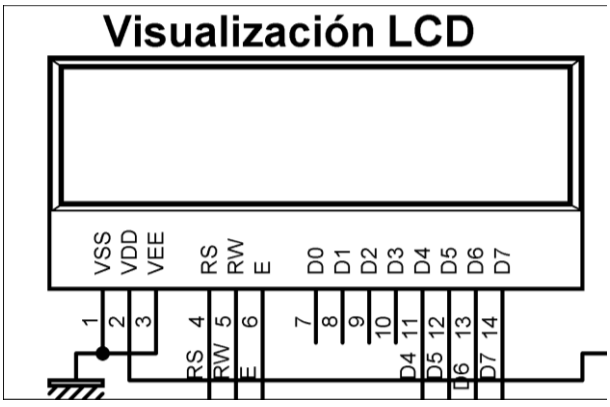


Figura 9. *Visualizador LCD*

Display 7 segmentos

Se disponen de 4 displays de 7 segmentos de ánodo común conectados a un controlador 4543 con sus ánodos conectados a masa través de un transistor de conmutación tipo 2n2222. De esta forma tenemos 4 bases de los transistores conectados a los pines D0-D3 a través de una resistencia limitadora de 1 KΩ y las cuatro entradas del decodificador, conectadas a los pines D4-D7.

Por lo tanto, solo se puede utilizar un tipo de visualización ya que comparten las mismas líneas de comunicación, así que deberemos de asegurarnos de haber activado el dispositivo de visualización mediante los 2 jumpers de 3 contactos disponibles, el comentado anteriormente que conmuta la alimentación y un segundo jumper que conecta el pin BL del decodificador a masa o a alimentación, para evitar así que los displays sigan activos aún quitándole la alimentación del controlador.

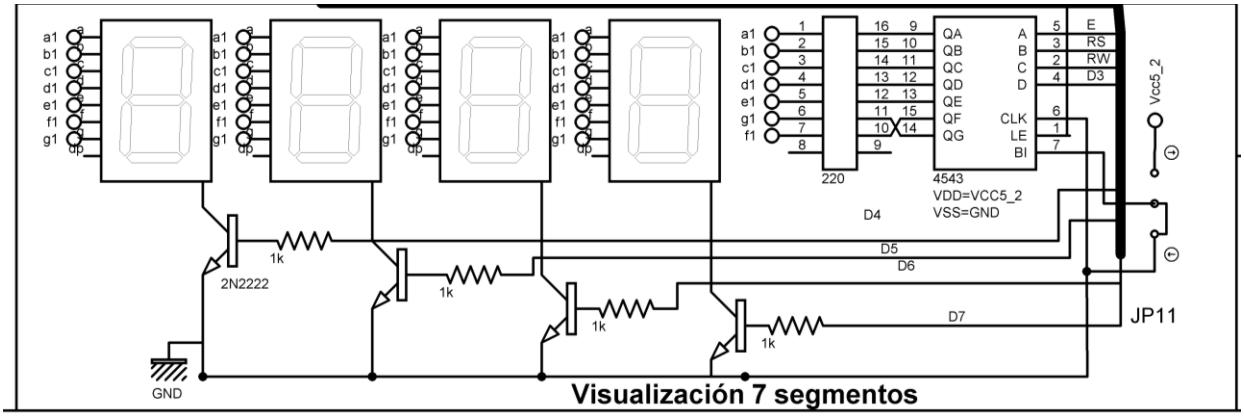


Figura 10. *Visualización 7 segmentos*

RTC DS1307

Se dispone de un reloj calendario en tiempo real integrado en la placa. Éste se comunica con el microcontrolador a través de un bus I2C, con lo cual 2 de sus patillas están conectadas a las líneas de transmisión SCL y SDA y necesitamos otras dos para la alimentación de 5V, por lo tanto, sabiendo que este es un integrado de 8 pines pasaremos a explicar la función de los otros 4. Los pines 1 y 2 deben ser conectados a un oscilador externo de 32768 Hz necesario para el funcionamiento del integrado. Los pines 3 y 7 son pines para funciones auxiliares, siendo la función del primero dotar al integrado de una alimentación ininterrumpida al conectarle una pila de entre 2 y 3,5 V, permitiendo así al integrado almacenar los datos al cortarse la alimentación. El pin 7 nos proporciona una salida cuadrada programable en colector abierto, pudiéndola utilizar para diferentes acciones, como encender un led cada segundo, interrumpir al microcontrolador cada intervalo de tiempo deseado, etc. En caso de no utilizar alguna de estas dos señales deberán conectarse a masa.

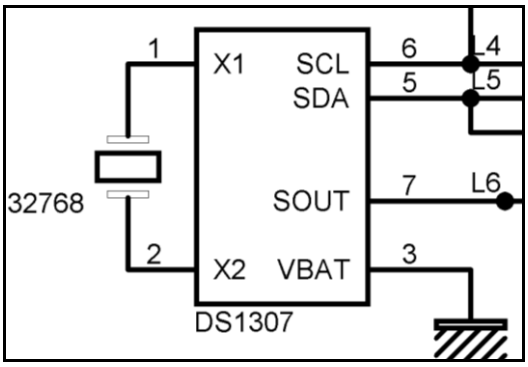


Figura 11. *RTC DS1307*

Convertidor AD/DA PCF8591

Este integrado se compone de un convertidor A/D de 4 entradas y un D/A con 1 salida y al igual que el reloj en tiempo real se comunica con el microcontrolador mediante el bus I2C. Las conexiones de este integrado son muy sencillas, es un dispositivos de 16 pines y disponemos, igual que en el caso anterior, de los pines de alimentación (8 y 16) y los de comunicación (9 y 10), además tenemos las 4 entradas analógicas (1-4) y la salida analógica (15). Las demás conexiones son para la configuración del integrado, 3 para la configuración de la dirección del integrado (5, 6 y 7) que se conectan a alimentación o a masa para generar la dirección del dispositivo permitiendo conectar varios dispositivos, además de una entrada o salida de oscilador (11), una masa analógica (13), selector de oscilador interno/externo (12) y tensión de referencia (14) que suele aplicarse mediante una resistencia de 2k2 conectada a alimentación polarizando un zener de 2,5 V.

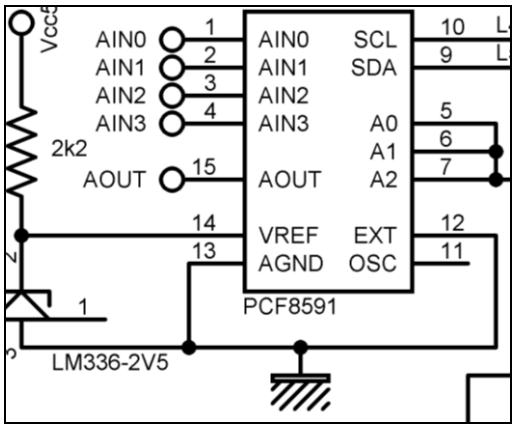


Figura 12. *PCF8591*

Comunicación RS232

Para dotar a la tarjeta de comunicación externa con la computadora se ha utilizado la comunicación RS232 a través de un conector DB9. Para ello, se ha utilizado el integrado MAX232 que con la ayuda de 4 condensadores externos es capaz de adaptarnos las señales TTL proporcionadas por el microcontrolador a aquellas necesarias para este tipo de comunicación, permitiéndonos conectar 4 lineas distintas, es decir, no únicamente las señales de transmisión como TxD y RxD sino alguna extra como pueden ser las de control de flujo (CTS y RTS) como es el caso. El esquema de conexión de este integrado se puede observar en la imagen y no necesita mayor explicación ya que únicamente necesita de condensadores electrolíticos.

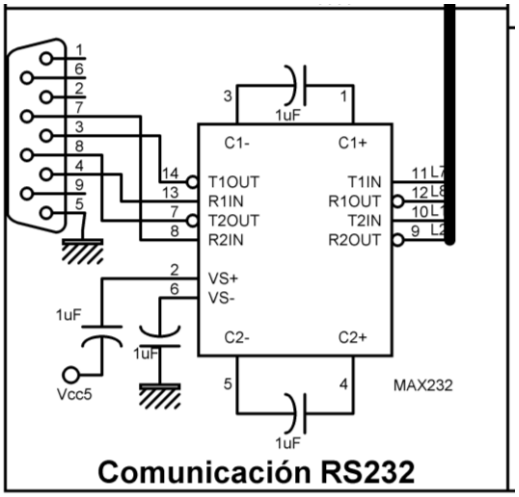


Figura 13. *MAX232*

Comunicación USB

Como alternativa a la comunicación RS232 con la computadora se ha añadido un conector USB clase B el cual dispone de 4 pines, dos de alimentación (Vcc y GND) y otros dos de datos, que se encuentran conectados directamente al microcontrolador (D+ y D-). Debemos tener en cuenta que el microcontrolador PIC16F877A con el que se trabaja por defecto en todas las prácticas no dispone de este tipo de comunicación para la cual deberemos utilizar otro microcontrolador como por ejemplo el PIC18F4550. Se ha incluido este tipo de comunicación debido al aumento de ventas de los ordenadores portátiles los cuales no incluyen puerto del tipo serie y deberemos utilizar comunicación USB.

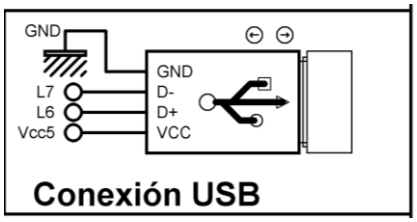


Figura 14. *Conexión USB*

Conexiones externas

En lo referente a la ampliación de la placa se ha destinado un conector DIL-20 en el que se encuentran las señales de alimentación (Vcc2 y GND) así como el puerto B al completo, las entradas y salidas analógicas del PCF8591 y el puerto E. Se ha utilizado un conector de 20 pines en vez de 40 el cual nos permitiría extraer todos los puertos del microcontrolador para reducir el espacio necesario tanto por el propio conector como el necesario para mover las pistas hasta él, además se ha tenido en cuenta que todos los demás pines puedan ser extraídos al exterior de otras formas distintas, sea extrayendo el componente y utilizando su conector SIL (caso del puerto D en el LCD) o a través de jumpers como en el puerto A. De este modo todos los puertos son accesibles desde el exterior de una forma más o menos rápida.

También se han dispuesto 5 bornes atornillados de dos contactos cada uno con las siguientes señales:

- Señal de alimentación externa (Vcc2 y GND)
- Contacto Relé
- Contacto Relé estado sólido (SSR)
- Señales PWM
- Entrada analógica
- Salida analógica

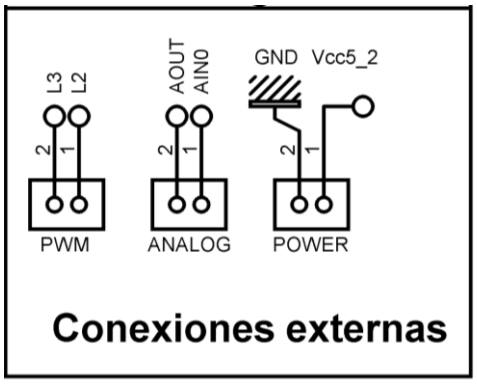


Figura 15.
Conexiones externas

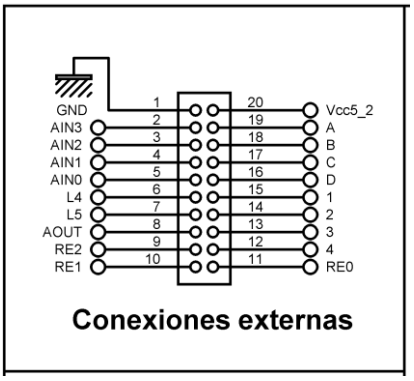


Figura 16. *Conector DIL20*

Tarjeta ampliación

Esta tarjeta externa se conecta a nuestra placa entrenadora mediante un conector DIL20 y permite ampliar nuestro circuito con dos funciones añadidas, que son:

- Sistema RC de segundo orden para simular el control de temperatura de una sala.
- Buzzer con transistor mosfet de activación para el control de frecuencias sonoras.

Además, dispone de un jumper para la conexión del pin USB a masa en el caso de querer utilizar la comunicación USB.

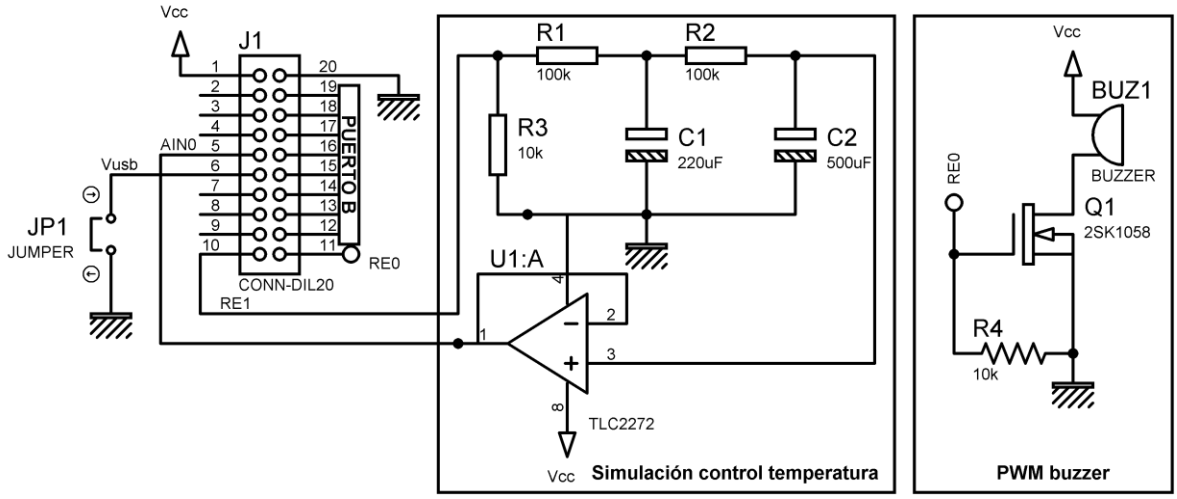


***Figura 17**. Circuito tarjeta externa*

Módulo ampliación relés

Este módulo se conecta a la alimentación mediante un conector con 2 polos y tierra y facilita el trabajo con los relés de la entrenadora. Dispone de dos enchufes conectados directamente a tensión para enchufar nuestra placa o cualquier otro aparato deseado, osciloscopio, fuente de alimentación externa. Además, incluye un tercer enchufe controlado por uno de los relés y una bombilla de señalización roja con conexión directa a otro relé. El módulo se conecta a a los relés mediante 4 conectores a presión que facilitan la inserción

.

Figura 18. *Módulo Relés*

Figura 19*. Detalle conexiones*

II.1.5 Circuito Final

En el siguiente apartado se muestra el esquema correspondiente al circuito final implementado, en el se encuentran recogidos todos los bloques comentados en el apartado anterior.

Para una mejor organización de los bloques los integrados no se encuentran unidos mediante líneas sino que existen etiquetas entre los diferentes puntos de unión.

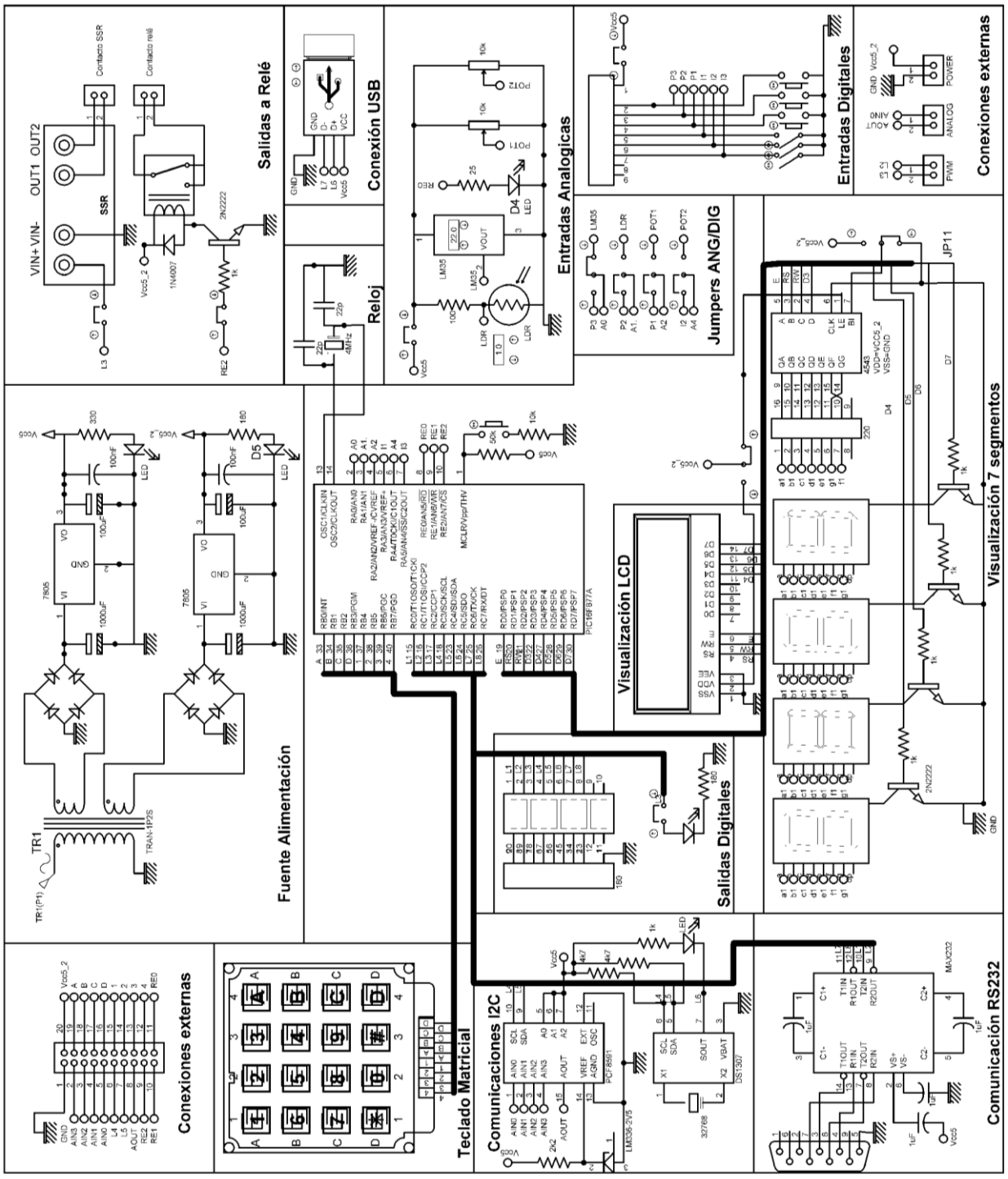


***Figura 20**. Circuito Final*

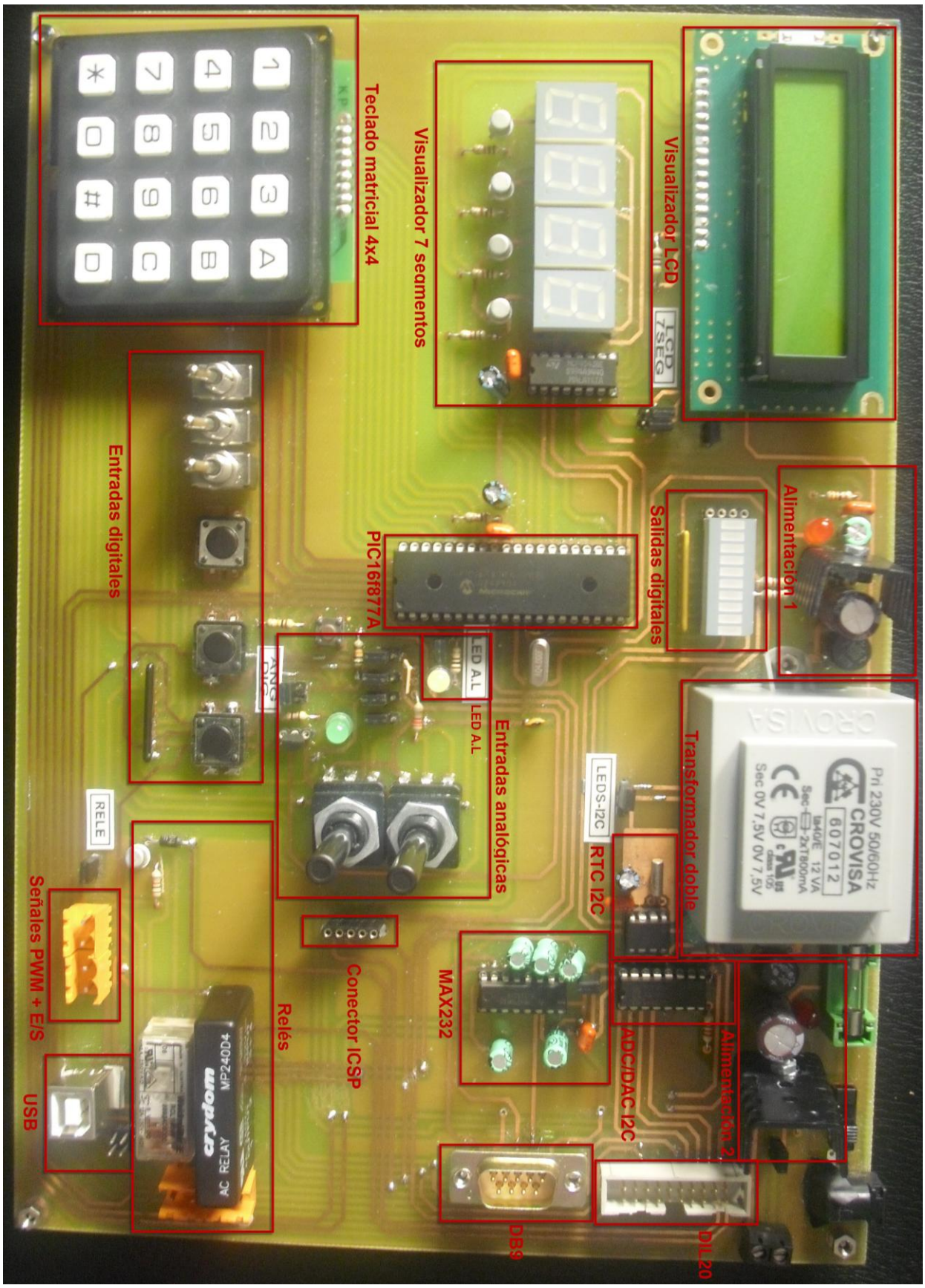


Figura 21. *Detalle circuito*

II.1.6 Conexiones

Una vez se han definido todos los aspecto y se ha observado el aspecto final del circuito podemos afirmar que todos los componentes están conectados con el microcontrolador mediante pistas de la PCB por lo que no deberemos realizar ningún tipo de conexión entre éstos para su funcionamiento. De todas formas, conviene pararse a observar el esquema del circuito y de los diferentes integrados para comprender correctamente su funcionamiento. Además, a la hora de trabajar con periféricos que comparten el mismo puerto, siendo el caso, por ejemplo, del visualizador LCD y los displays de 7 segmentos, deberemos configurar los jumper adecuados para cada función, por ello a continuación se recoge una lista con los principales jumper de la placa y su función, esta lista no sería del todo necesario puesto que la función de cada jumper se encuentra rotulada en la placa pero servirá así como recopilatorio y para referencias posteriores. De izquierda a derecha tenemos:

1. LCD/7SEG: Formado por dos jumper situados en la esquina inferior izquierda del módulo LCD. Situando ambos jumper en su posición superior activamos el módulo LCD y en la inferior el controlador 7 segmentos.
2. LED alta luminosidad. Jumper situado al lado del led que lo conecta o desconecta del microcontrolador.
3. Entrada Analógicas/Digitales. Formado por 4 jumpers que seleccionan una a una las 4 entradas. En su posición superior conectan las entradas analógicas y en la inferior las digitales. Sus funciones son, de izquierda a derecha:
   - LM35/P1
   - LDR/P2
   - POT1/P3
   - POT2/I2
4. Alimentación analógicas/digitales. Formado por dos jumpers situado delante del LM35 que se conecta horizontalmente. Conectando los dos pines superiores se alimentan las entradas analógicas y mediante los inferiores se alimentan las digitales.
5. I2C/leds. Jumper situado delante del transformador. En su posición izquierda alimenta a la barra de leds y en su posición derecha los periféricos de comunicación I2C y el integrado MAX232.
6. Conexión relés. Dos jumpers de dos contactos situados cada uno delante de su relé correspondiente conectan o desconectan éstos al microcontrolador.

II.1.7 Proceso de grabación

Una vez definido el hardware que definirá el aspecto final de la placa pasaremos a hablar sobre el programador de nuestra placa y del software necesario para su programación.

Circuito Programador

Lo primero que se debe remarcar en este aspecto, como se ha podido observar, viene referido a la falta de un programador integrado en la placa, y esto es así porque se han dispuesto los elementos pasivos necesarios para su programación mediante ICSP y se ha dejado libre un conector de 5 pines disponible para tal fin.

De esta forma se deja al usuario la libertad de elección del programador pudiendo así utilizar aquel del que ya disponga de antemano, o la posibilidad de comprar uno comercial así como de montar uno propio como el que se definirá en las siguientes páginas. De esta forma se deja libertad al usuario de utilizar programadores de diferentes características y por lo tanto también de diferentes precios según si dispone de un PC con puerto paralelo o serie, donde podemos encontrar programadores más económicos, o por el contrario dispone de un ordenador portátil sin estos puertos y necesita programar mediante el puerto USB. En relación con este último aspecto se escribe el siguiente punto, en dónde se definen algunas de las características comerciales del programador comercial PicKit 2 de la marca Microchip el cual se ha utilizado para la programación de la placa durante el desarrollo de este proyecto.

Programador PICkit

El PICkit 2 es un programador para microcontroladores PIC de la marca microchip, es decir, la misma empresa que fabrica estos controladores. Es un programador que funciona por el puerto USB y está situado en el rango de los programadores de bajo coste. Además de programador nos proporciona la utilidad de debugger. A continuación se destacarán algunas de sus características más importantes.

- Como programador el PICkit 2 es capaz de programar la mayor parte de los microcontroladores flash de microchip:
    - Baseline: PIC10F, PIC12F5xx, PIC16F5xx).
    - Midrange: PIC12F6xx, PIC16F, PIC18F, PIC24, dsPIC30 y dsPIC33.
    - Variedad de EEPROM de microchip

Como Debugger el PICkit 2 se puede utilizar para realizar el "in-circuit debugging", con lo cual el usuario podrá comprobar y modificar el programa con el microcontrolador integrado en el hardware.

Figura 22. *Programador PICkit 2*

Programa PICkit 2

El programa grabador PICkit 2 viene incluido con el programador PICkit 2 y se puede descargar gratuitamente desde la página web de microchip.

Es un sencillo programa que nos permite grabar un archivo *.hex de nuestro ordenador al microcontrolador. Su pantalla de trabajo es sencilla y sus controles son intuitivos así que no se necesita una explicación detallada para comenzar a trabajar con él, el programa reconocerá automáticamente si el PICkit2 y el microcontrolador están conectados, nos reconocerá el modelo de nuestro microcontrolador y lo único que deberemos hacer será seleccionar el botón "Auto import Hex+ Write Device" situado en la parte inferior derecha de la pantalla, de esta forma el programa se encargará de copiar el archivo que seleccionemos al microcontrolador, además esta tarea se repetirá cada vez que modifiquemos el programa y compilemos desde nuestro software (MPLAB, CCS).

Por lo tanto, el usuario únicamente debe conocer las 3 pantallas de trabajo posible para iniciarse con este programa. En la primera no encuentra el PICkit 2 y como nos dice deberemos comprobar la conexión USB e ir a Tools->Check Communication, en la segunda no encuentra el microcontrolador y deberemos comprobar el estado de nuestro microcontrolador y sus conexiones. Por último, la tercera pantalla es la de correcto funcionamiento y nos indica el tipo de microcontrolador y el programa cargado en él, pudiendo leer o escribir sobre él.

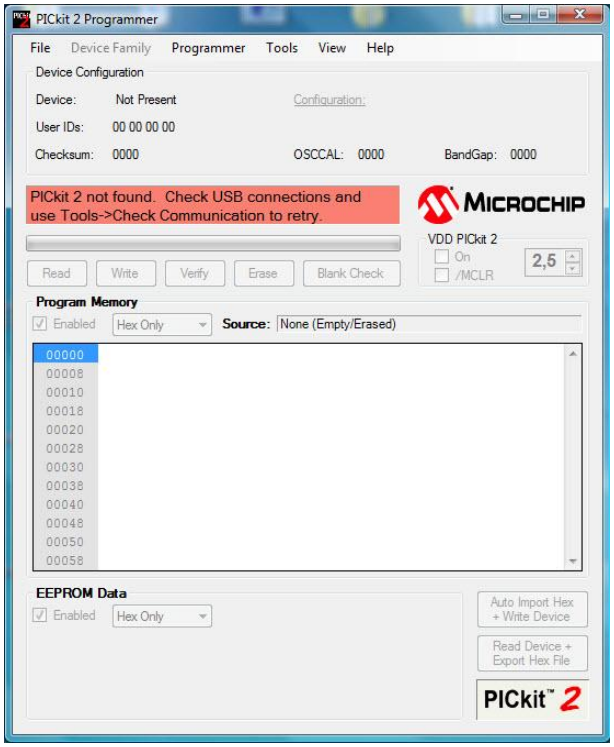


Figura 23. *PICkit no conectado*

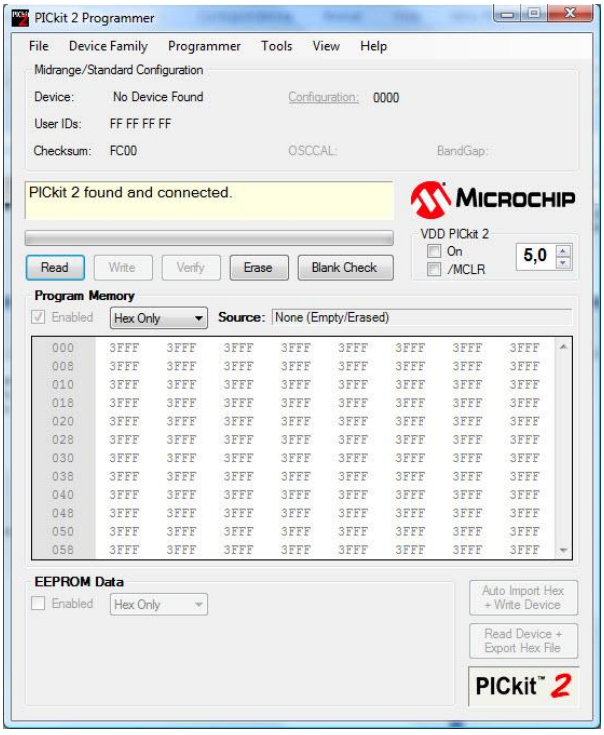


Figura 24. *Microcontrolador no conectado*

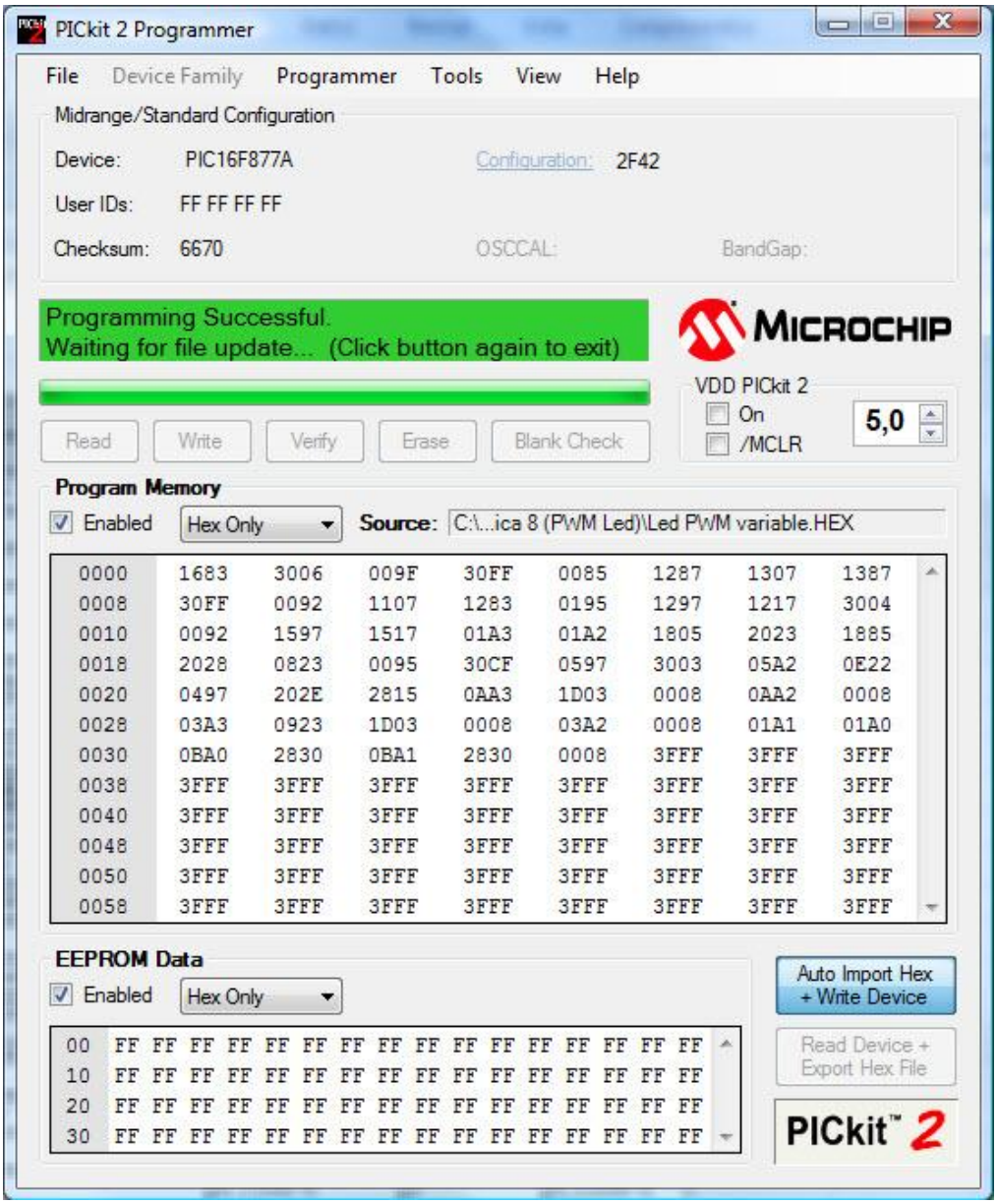


Figura 25. *Conexión correcta*

Programador JDM

El programador JDM, también conocido como PIC-Programmer 2, es un programador muy utilizado debido principalmente a su sencillo montaje y buena respuesta de programación. Fue desarrollado por Jens Dyekjær Madsen (JDM) entre 1996 y 1998 y algunos programadores actuales están basados en este modelo, como el programador comercial TE-20. Para la programación se utiliza el programa IC-Prog seleccionando la opción JDM Programer.

Una de las ventajas que nos ofrece este programador es que no necesita de fuente externa de alimentación ya que la extrae del puerto serie de la misma forma que las señales de programación. En la actualidad, existen una gran cantidad de variantes de este circuito el cual es libre para poder modificarlo y recrearlo.

Por todo lo comentado anteriormente, se recomienda al lector que en caso de necesitar un programador de microcontroladores PIC se plantee la implementación del programador que se muestra en la siguiente figura ya que tiene un coste bajo y su montaje es sencillo. Se debe tener en cuenta que los ordenadores portátiles no cuentan con puerto de comunicaciones serie (DB9) y los convertidores USB-RS232 no son compatibles con estos programadores. Por lo tanto los propietarios de ordenadores portátiles deberán adquirir o implementar un programador que funcione a través de USB.

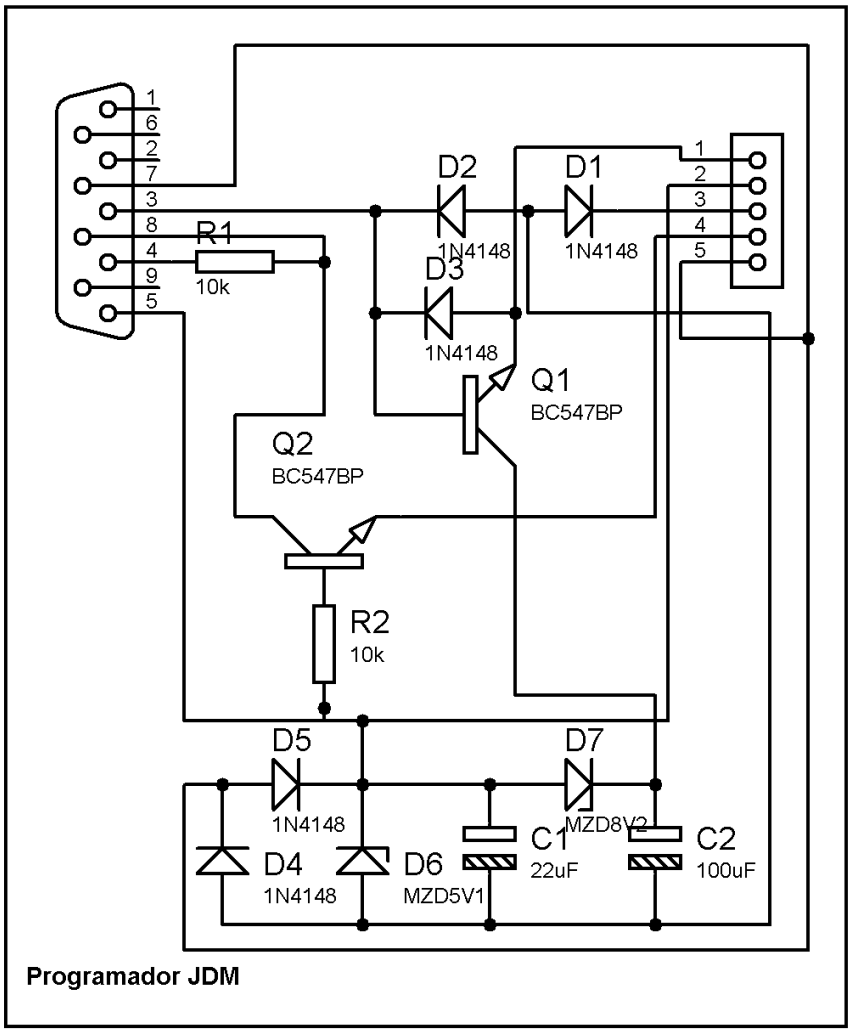


Figura 26. *Esquema programador JDM*

A la hora de conectar un programador mediante ICSP a nuestro circuito deberemos asegurarnos de cumplir una serie de recomendacioens para conseguir una correcta programación y evitar posibles daños en nuestro circuito o programador. Las principales recomendaciones a tener en cuenta són:

- No se deben conectar circuitos activos al pin /MCLR.
- Si /MCLR es usado para resetear el PIC se debe conectar un resistor más grande de 56kΩ entre /MCLR y *Vdd*.

- Las señales de programación deben estar libres de cargas capacitivas e inductivas.
- Cuando el voltaje operativo Vdd está controlado por el módulo de programación puede ser necesario aislar el pin Vdd del PIC del resto del circuito. Este aislamiento puede ser realizado conectando la alimentación del Pic a través de un diodo Schottky.
- Para mantener el pin PGM en *estado bajo* durante la programación debe conectarse a masa con un resistor de entre 2,2kΩ y 10kΩ.
- Siempre activar el "Power-Up Timer" en la palabra de configuración, porque produce un retardo de más de 40ms que da suficiente tiempo para alcanzar un *Vdd* estable antes del inicio de cualquier operación en el PIC, y evita la ejecución no deseada del programa antes de entrar en el modo de programación.
- Durante la programación los pines del microcontrolador se encuentran en estado de alta impedancia, el circuito debe tolerar esta situación y interferir lo menos posible. En caso de que fuera necesario se puede conectar resistores entre los diferentes pines y la alimentación del circuito.

Programa IC-Prog

El software programador IC-Prog realizaría la misma función explicada anteriormente para el PICkit 2 pero de una forma más genérica para una gran variedad de programadores comerciales y otros de fabricación propia. Este programa se puede descargar gratuitamente desde su web (www.ic-prog.com) y funciona sobre la plataforma Windows.

La primera vez que ejecutemos el programa nos aparecerá una pantalla como la siguiente en la que se nos indica que és la primera vez que ejecutamos el programa y debemos configurar algunos aspectos de nuestro harware.

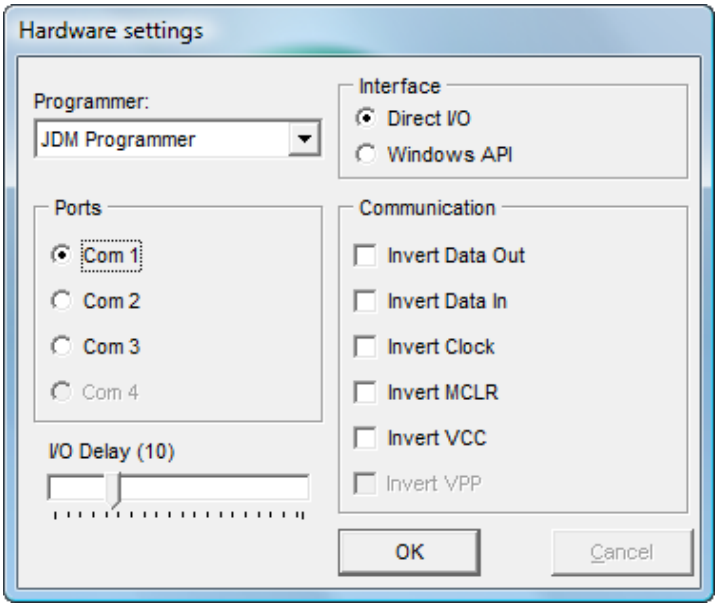


***Figura 27**. Pantalla inicio*

En esta pantalla deberemos configurar algunos parámetros como son:

- Seleccionar el tipo de programador: JDM Programer
- Tipo de interfaz: Windows API
- Seleccionar invertir MCLR y VCC

Una vez configurado el harware a utilizar se inicializará el programa y nos aparecerá una pantalla como la siguiente:

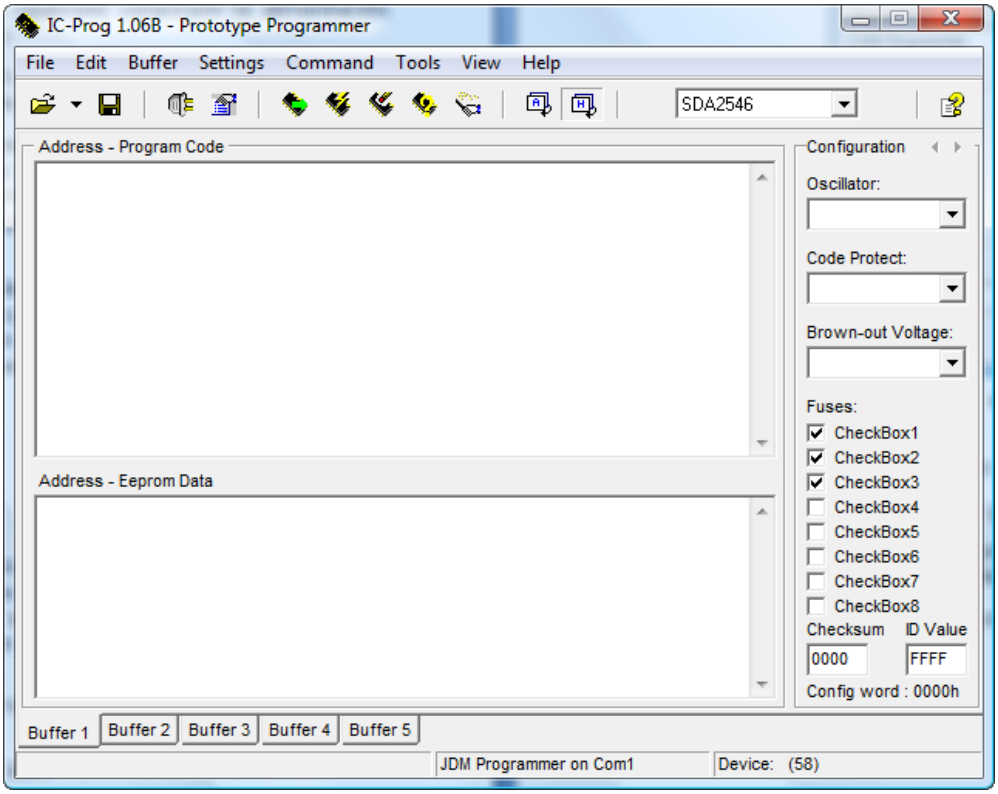


Figura 28. *Pantalla principal*

Ésta es la pantalla principal y desde ella configuraremos las diferentes opciones del programa y programaremos nuestro microcontrolador. El programa tiene una gran variedad de funciones configurables pero se pasará a explicar la programación y se deja en manos del usuario el consultar los tutoriales que se encuentran en la web para aprender más funciones del programa.

El proceso de programación es muy sencillo y consiste principalmente en 4 pasos:

- Elección del dispositivo a programar: en este caso el PIC16F877A
- Seleccionar el archivo a grabar: Archivo->Abrir Archivo
- Definición bits de configuración: Es preferible hacerlo desde el ensamblador o compilador activándolos éste automáticamente.
- Programación del microcontrolador: Comando->Programador Todo.

II.2 Definición de componentes

En el siguiente apartado se definirán los aspectos principales de los diferentes componentes utilizados en éste laboratorio de microcontroladores para así hacer éste proyecto más accesible a usuarios con pocos conocimientos de electrónica, facilitando así una mayor comprensión del hardware y una familiarización con la placa diseñada.

Leds

Se entiende como LED (Light-Emitting Diode) o diodo emisor de luz un dispositivo semiconductor (diodo) que emite luz incoherente y de espectro reducido cuando se polariza de forma directa lo que hace circular una corriente a través de él. Su color puede variar según el material semiconductor utilizado en su elaboración. En la siguiente tabla se recogen los distintos tipos de diodos así como su tensión y corriente consumidas:

Tabla 1. Características leds

Color	Corriente	Tensión	Luminosidad
Rojo	10 mA	2,4 V	12,5 mcd
Verde (L5V)	10 mA	2,4 V	4 mcd
Blanco (alta lum.)	20 mA	3 V	16 cd
Barra leds	25 mA	2.2 V	9,5 mcd

En nuestro caso se ha utilizado una barra de leds que integra 10 LEDs de forma rectangular en un único encapsulado, lo que mejora su estética y reduce el espacio necesario.

Figura 29. *Barra de LEDs*

Interruptores y pulsadores

Estos dispositivos no necesitan de demasiada explicación ya que son conocidos por todos y los utilizamos diariamente. Son dispositivos cuya funciones la de controlar el flujo de la corriente, es decir, permitir o no permitir el paso de ésta, o desviarla hacia diferentes caminos como el caso de los conmutadores. Aunque su funcionamiento es conocido por la mayoría se aprovecha se presentan a continuación unas imágenes de los dispositivos utilizados en ésta placa ya que posiblemente no estamos tan acostumbrados a verlos y pueden dificultar su localización.

***Figura 30**. Interruptor*

***Figura 31**. Pulsador*

Potenciómetros

El funcionamiento de un potenciómetro es muy sencillo ya que se basa en la variación de resistencia mediante una acción mecánica pudiendo ser ésta lineal o circular como en nuestro caso. En el potenciómetro utilizado y en lo general en la mayoría de los potenciómetros dispondremos de 3 patillas, entre las dos exteriores dispondremos de una resistencia fija de valor igual al nominal del potenciómetro, y es entre la patilla del medio y una de las exteriores en donde tendremos las variación de resistencia, siendo positiva o negativa dependiendo de la patilla seleccionada.

***Figura 32.** Potenciómetro*

Teclado 4x4

Éste es un dispositivo ampliamente utilizando en circuito electrónicos para la entrada de datos tanto numéricos, como simbólicos y alfabéticos. El teclado posee 16 teclas interconectando filas y columnas y dispone de un conector SIL8 con las 4 filas y 4 columnas. Cabe considerar que existen teclado de 4x3 y otros tamaños para distintas aplicaciones.

***Figura 33**. Teclado matricial 4x4*

Displays 7 segmentos (4543)

Los visualizadores o "Displays" de 7 segmentos están compuestos por siete LEDs (en realidad ocho, porque usualmente cuentan con un punto decimal), dispuestos de tal forma que representan la imagen de un número ocho, de esta forma, activando los diferentes segmentos podemos representar todos los números. Los segmentos se encuentran unidos por su ánodo o su cátodo, reduciendo así su número de patas a 10 y diferenciando así dos familias de displays: ánodo común y cátodo común.

En los visualizadores de ánodo común todos los ánodos de los segmentos se encuentran unidos en una patilla que debe ser conectada a potencial positivo. El encendido de cada segmento se realiza aplicando potencial 0 por la patilla correspondiente a través de una resistencia que limite el paso de la corriente.

En los visualizadores de cátodo común todos los cátodos de los segmentos se encuentran unidos en una patilla que debe ser conectada a potencial 0. El encendido de cada segmento se realiza aplicando potencial positivo por la patilla correspondiente a través de una resistencia que limite el paso de la corriente.

El integrado 4543 es un decodificador BCD/7segmentos que nos permite controlar un display de 7 segmentos utilizando únicamente 4 líneas de nuestro microcontrolador, indicando por éstas el código BCD del número a mostrar.

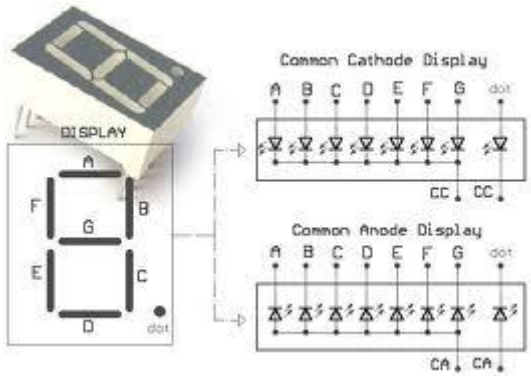


Figura 34. *Display 7 segmentos*

Módulo LCD

Las pantallas de cristal líquido o LCD son dispositivos capaces de mostrar cualquier carácter alfanumérico, siendo pantallas de bajo coste que las hace muy populares en circuitos donde se necesita visualizar algún tipo de información de forma sencilla y económica.

Estas pantallas están formadas por una matriz de caracteres formados por puntos, (normalmente 5x7 o 5x10), divididos en filas y columnas, donde algunos de los más comunes son (2x16, 4x40,…). La visualización de estas matrices es controlador por un microcontrolador incorporado en el módulo LCD, el estándar es el Hitachi 44780, siendo la mayoría de librerías dedicadas a este microcontrolador, aunque existen una gran variedad de módulos LCD en el mercado que utilizan microcontroladores iguales o compatibles.

Figura 35. *Módulo LCD*

RTC DS1307

En la placa se ha incluido un bus I2C para comunicar periféricos capaces de trabajar con este tipo de bus para comunicarse con el microcontrolador. Uno de estos periféricos incluidos y uno de los más conocidos por los usuarios que se inician con éste tipo de comunicación es el reloj en tiempo real RTC DS1307. Éste dispositivo es capaz de informar de la hora y fecha real al microcontrolador dotándolo de una visión "real" del tiempo en el mundo exterior muy útil para las aplicaciones de adquisición de datos. Este dispositivo es muy útil para la iniciación con este tipo de bus de comunicaciones ya que en internet se dispone de infinidad de librerías ya diseñadas que facilitarán el trabajo con el reloj.

***Figura 36**. DS1307*

AD/DA PCF8591

El integrado PCF8591 es un dispositivo que trabaja en bus I2C y por lo tanto, utilizando solo dos líneas del microcontrolador nos proporciona:

- Convertidor Digital-Analógico (DAC) de 8 bits.
- Convertidor Analógico-Digital (ADC) de 8 bits.

Por lo tanto, es un dispositivo que nos permite ampliar las patillas de medición analógica con las cuatro entradas que dispone el integrado, así como añadir una salida analógica de 0 a 5 V.

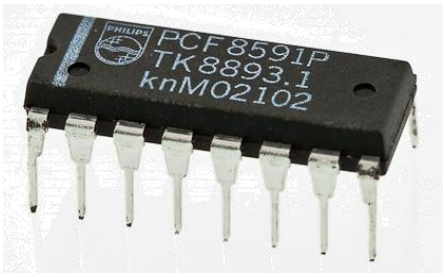


***Figura 37**. PCF8591*

MAX232

El integrado max232 nos permite adaptar las señales del TTL a lo estándares de comunicación RS232, permitiendo así, la comunicación entre nuestro circuito y la computadora. Para ello, éste integrado se ayuda de una serie de componentes pasivos externos (condensadores) y genera una tensión que llega a los 12V, proporcionándonos dos canales de recepción y dos de transmisión de datos.

***Figura 38**. MAX232*

Relé

El relé es un dispositivo que consta de dos circuitos diferentes: uno electromagnético (electroimán) y uno de contactos al que conectamos el circuito que queremos controlar. Su funcionamiento se basa en el fenómeno electromagnético: cuando la corriente atraviesa la bobina, produce un campo magnético que magnetiza el núcleo de hierro dulce (ferrita), el cual atrae al inducido lo que hace que los contactos se toquen, volviéndose a separar automáticamente cuando la corriente se desconecta.

A continuación se analizarán sus características técnicas para conocer un poco más su funcionamiento.

- Bloque electromagnético
    - Corriente de excitación. Intensidad necesaria a través de la bobina para poder activar el relé.
    - Tensión nominal. Tensión de trabajo a la cual se debe alimentar el relé para que éste se pueda activar.
    - Tensión de trabajo. Margen existente entre las tensión máxima y mínima indicada por el fabricante para que el relé puede funcionar correctamente.
    - Consumo nominal de la bobina. Potencia que consume la bobina cuando el relé está excitado con la tensión nominal a 20 ºC.
- Contactos o parte mecánica
    - Tensión de conexión. Tensión en los contactos antes de cerrar o después de abrir.
    - Intensidad de conexión. Corriente máxima que un relé puede conectar o desconectar.
    - Intensidad máxima de trabajo. Intensidad máxima que puede circular por los contactos cuando se encuentran cerrados

***Figura 39**. Relé electromagnético*

SSR

Los relés de estado sólido o SSR (Solid State relay) son dispositivos que dejan de banda los contactos metálicos utilizados en los relés clásicos y utilizan para tal fin transistores, tiristores o triacs permitiendo controlar elevadas cargas de potencia a partir de señales de bajo voltaje e intensidad.

Estos dispositivos consisten en un circuito electrónico formado por un disparador por nivel acoplado a un interruptor semiconductor (transistor o tiristor), siguiendo la siguiente estructura:

- Circuito de entrada/control:
    - Tensión continua: el circuito de entrada suele estar formado únicamente por un LED/Fotodiodo o acompañado de una resistencia en serie, pudiéndolo encontrar también con un diodo en anti paralelo para evitar la inversión de la polaridad. Los niveles de entrada son compatibles con TTL, CMOS además de otros valores estandarizados como 12V, 24V,…
    - Tensión alterna: El circuito de entrada es similar al anterior con la incorporación de un puente rectificador y una fuente de corriente continúa para polarizar el diodo LED.

- Acoplamiento. El acoplamiento con el circuito usualmente se realiza mediante un opto acoplador que se encuentra conectado con el circuito de disparo del Triac, proporcionando así un aislamiento entre éste y la entrada del relé.

- Circuito de Salida/Conmutación. En éste bloque del relé se encuentran recogidos los semiconductores de potencia así como su correspondiente circuito excitador. Como ya se ha comentado éste puede ser un triac, tiristor, transistor,… variando según queramos conmutar corriente continua o alterna.

Este tipo de relés nos ofrecen una serie de ventajas con respecto a los relés electromecánicos como pueden ser:

- Menor peso y ruido.
- Mayor velocidad de trabajo permitiendo conmutar a mayores frecuencias, permitiendo así hacer controles PWM.
- Inmunidad a choques y vibraciones
- Capacidad de conmutar altas corrientes y voltajes sin producir arcos
- Varios kilovoltios de aislamiento entre entrada y salida.

También se deben considerar algunas desventajas de éstos dispositivos que pueden hacer que no sean útiles para alguna de nuestras aplicaciones:

- Mayor coste que los relés clásicos.
- Dispositivos de una sola posición, no pudiendo conmutar a la vez varias cargas independientes.

Figura 40. *Relé SSR*

LM35

El LM35 es un sensor de temperatura encapsulado con rango de medida comprendido entre -55º y +150ºC. El encapsulado más común es el to-92, idénticos a transistores como el 2n2222, y que dispone de 3 patas, 2 para la alimentación y otra en donde nos proporciona directamente un valor de tensión proporcional a la temperatura medida, siendo esta salida lineal y con relación mV/ºC.

Figura 41. *LM35*

LDR

Una LDR (Light Dependent Resistor), resistencia dependiente de la luz, es una resistencia cuyo valor varía en función del nivel de luz que incida sobre ella, siendo su resistencia menor contra más luz incida sobre su superficie.

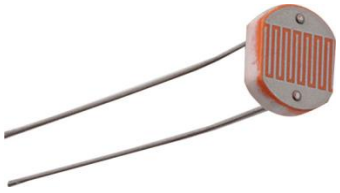


Figura 42*. LDR*

Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Anexo III.
Manual de Prácticas

"DISEÑO DE UNA PLATAFORMA DOCENTE PARA EL APRENDIZAJE DE MICROCONTROLADORES 'PIC' DE MICROCHIP"

PFC presentado para optar al título de Ingeniero Técnico Industrial especialidad ELECTRÓNICA INDUSTRIAL

por **Víctor Bueno Álvez**

Barcelona, 12 de Enero de 2011

Tutor proyecto: Herminio Martínez García
Departamento de Ingeniería electrónica (DEEL)
Universitat Politècnica de Catalunya (UPC)

ÍNDICE PRÁCTICAS

Introducción

Hasta el momento nos hemos dedicado al estudio del hardware de la placa entrenadora. Por otro lado, debemos tener en cuenta otro aspecto muy importante a la hora del diseño de éste "kit" didáctico, el conjunto de prácticas con el programa para el microcontrolador y la explicación de la práctica oportuna.

El kit está formado por una serie de 12 prácticas en las que se trabajan los distintos periféricos de la tarjeta así como las distintas operaciones que nos permite el microcontrolador. Estas se encuentran organizadas en orden ascendente de dificultad para así hacer más fácil la familiarización con el microcontrolador. Además, podríamos dividir éstas en dos grupos bien diferenciados según el lenguaje de programación con el que se trabaja. Las primeras, más sencillas, están realizadas en ensamblador mediante el programa MPLAB IDE v8.36 proporcionado gratuitamente por microchip.

El lenguaje ensamblador permite al usuario una mayor comprensión del funcionamiento del programa debido a que cada instrucción de programa se corresponde con una instrucción del microcontrolador, por otra banda, al ir aumentando en dificultad, éste programa hace más pesada la programación y aumenta el tiempo de realización, por ello se utiliza un compilador en C en las últimas sesiones para aligerar un poco el trabajo, aunque sería igualmente factible su implementación en MPLAB. El compilador utilizado es el llamado PIC C Compiler, de la casa CCS, éste es un compilador muy utilizado por aficionados y profesionales debido a su simplicidad y fácil comprensión.

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 1 - Introducción a la programación en ensamblador
(Tutorial MPLAB)

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

Con esta primera práctica se pretende introducir al lector en el manejo del programa MPLAB de microchip. Para ello se realizará una breve introducción de las diferentes herramientas del programa y se procederá a la realización de un programa de forma completamente guiada para servir como referencia para prácticas posteriores.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa ensamblador MPLAB y el grabador PICkit2.

3. INTRODUCCIÓN MPLAB

El MPLAB IDE es una herramienta software de "Entorno de Desarrollo Integrado" (Integrated Development Enviroment, IDE) que se ejecuta bajo Windows. Con este entorno se pueden desarrollar aplicaciones para los microcontroladores PIC.

El MPLAB incluye todas las utilidades necesarias para la realización de proyectos con microcontroladores PIC, permite editar el archivo fuente del proyecto, además de ensamblarlo y simularlo en pantalla para comprobar cómo evolucionan tanto la memoria de datos RAM, como la de programa ROM, los registros SFR, etc., según progresa la ejecución del programa.

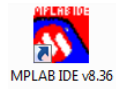
El MPLAB incluye:

- Un editor de texto que nos permite la escritura de nuestro código fuente.
- Un ensamblador llamado MPASM que traduce código fuente a lenguaje máquina para que pueda ser interpretado por otra computadora.
- Un simulador llamado MPLAB SIM, para simular en la misma pantalla nuestro programa y así poder corregir posibles errores.
- Un organizador de proyectos y otros. Herramientas que nos facilitan el trabajo con los diferentes programas que generemos.

4. DESARROLLO DE LA PRÁCTICA

Configuración MPLAB

Iniciamos el programa utilizando el icono que aparece en el escritorio, que será como el siguiente:



Una vez abierto el programa procederemos a crear una nueva página de programa a través del menú File>New, o con las teclas (Ctrl+N).

Antes de proceder a la escritura del código fuente deberemos configurar al MPLAB de acuerdo con el microcontrolador utilizado. Para ello vamos a Configure>Select Device y seleccionamos el microcontrolador deseado de la lista.

Además, es recomendable darle nombre al fichero y guardarlo al inicio para luego ir guardando lo cambios progresivamente. Para ello vamos a File>Save As… Se escribe el nombre de archivo deseado añadiendo la extensión .asm y se guarda en la carpeta deseada.

El proceso de ensamblado del programa así como algunas otras características que se consideren importantes se irán introduciendo a lo largo de la práctica.

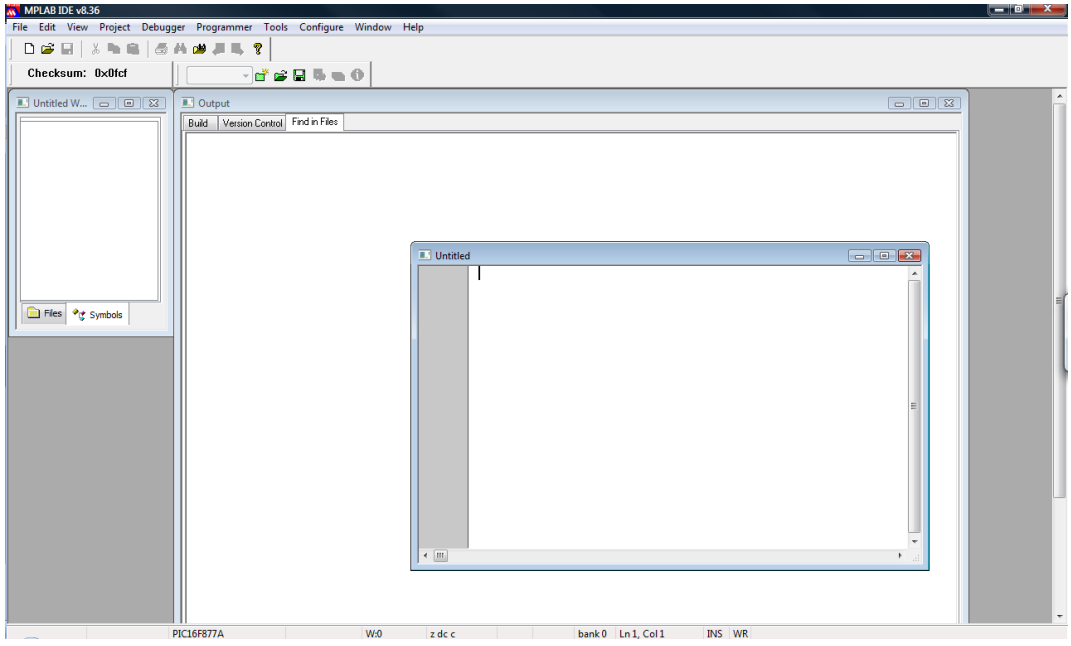


Figura 1. *Pantalla trabajo MPLAB*

Escritura del Programa

Una vez configurado el MPLAB procedemos a la escritura de nuestro propio programa. En nuestro caso utilizaremos un programa sencillo que se encarga de encender un led conectado al puerto C del microcontrolador con el objetivo de familiarizarnos con el MPLAB, el programa grabador y la tarjeta entrenadora.

El programa utilizado es el siguiente:

```
;*****************************************************************************************
;       Programa 1.1: Encendido Led (Hola Mundo)
;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
;       Autor: Víctor Bueno Alvez
;
;       Este programa eneciende un led conectado al pin 0 del puerto C.
;       Microcontrolador: PIC16F877A
;       Frecuencia: 4 MHZ
;*****************************************************************************************

;**********Zona de Definiciones**********
    __CONFIG _WDT_OFF&_PWRTE_ON&_HS_OSC&_LVP_OFF&_CP_OFF ; Configuración para el programador

    LIST p=16F877A
    INCLUDE <P16F877A.INC>

    ORG 0x00 ; Inicio de programa

;************Zona de programa************
Inicio
    bcf STATUS,RP0      ; Accede a
    bcf STATUS,RP1      ; banco 0
    clrf PORTC          ; Limpia PORTC
    bsf STATUS,RP0      ; Accede a banco 1
    clrf TRISC          ; Configura todos las patitas de PORTC como salidas
    bcf STATUS,RP0      ; Regresa a banco 0
Led
    bsf PORTC,0         ; La línea RC0 de PORTC toma el valor de 1, se enciende el LED
    goto Led            ; Va a la etiqueta Led

    END ;Fin de programa
```

Ensamblado del programa

Después de la escritura de todo el código del programa debemos proceder al ensamblaje del mismo para así poder generar los archivos que se grabarán en el microcontrolador.

Para ensamblar el programa iremos a Project>Quickbuild xxxxxx.asm, o con la combinación de teclas Alt+F10. Posteriormente se abrirá una ventana donde se podrá visualizar el proceso de ensamblado y el resultado final. El proceso de ensamblado puede dar como resultado un letrero que indique (BUILD SUCCEDED), lo que indica que el ensamblado se ha desarrollado correctamente con lo cual podemos pasar a su simulación o grabación en el microcontrolador. Aún con el mensaje de ensamblado correcto puede aparecernos una serie de mensajes en donde se nos

indican una serie de aspecto a tener en cuenta o que tienen posibilidad de causar errores aun no impidiendo el correcto ensamblado.

Debemos saber que el proceso de ensamblado no nos indica que el programa funcione correctamente, el no es capaz de entender nuestro propósito con ese programa, solo se encarga de mirar que la estructura y sintaxis del programa sea la correcta.

En el caso de que el programa contenga errores el proceso de ensamblado nos mostrará el letrero (BUILD FAILED) y un listado con los errores encontrados. Haciendo doble clic sobre estos errores nos dirigiremos directamente a la línea en donde éstos se encuentran.

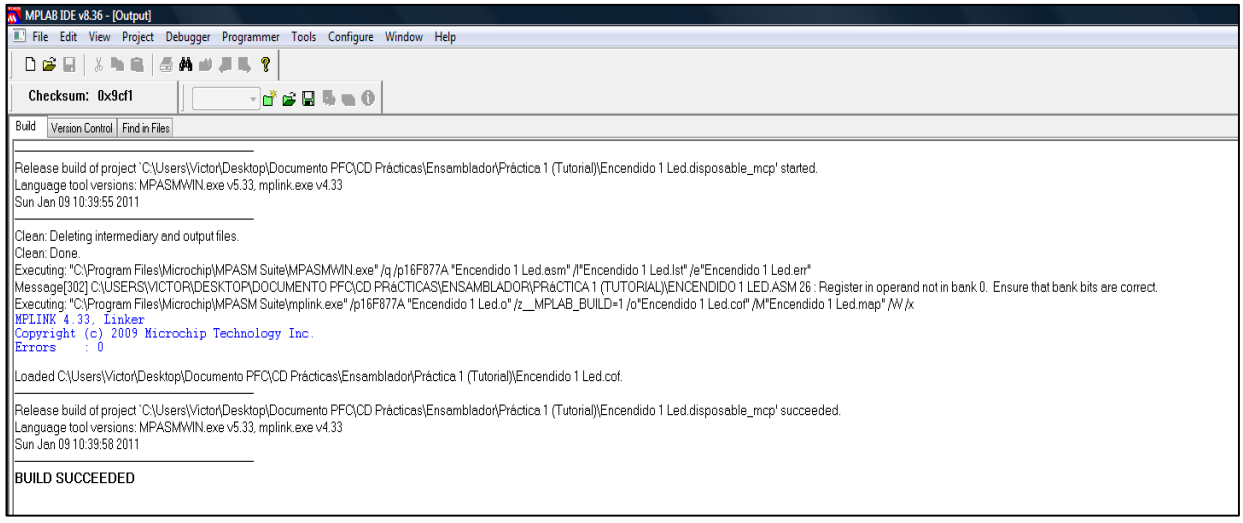


Figura 2. *Ensamblado correcto*

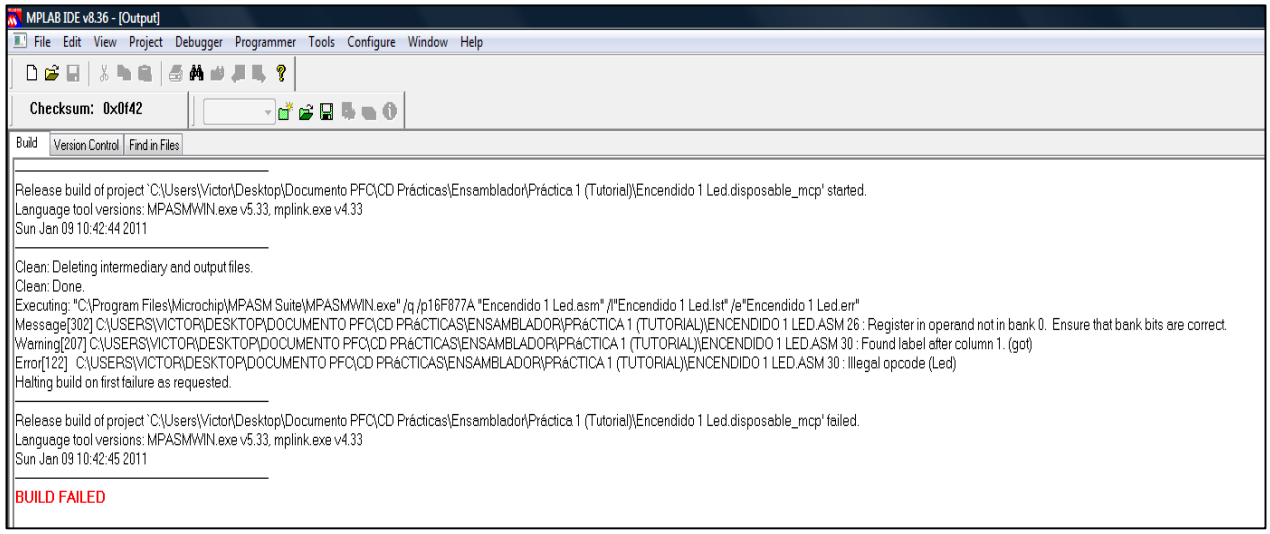


Figura 3. *Ensamblado fallido*

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 2 – Manejo de los puertos de E/S

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

Con esta práctica se pretende trabajar con los diferentes puertos de Entrada/Salida del microcontrolador. Con este objetivo se utilizarán 2 pulsadores conectados al puerto A del microcontrolador y 8 diodos led conectados al puerto C. De esta forma se remarcan 3 objetivos distintos en esta práctica con diferentes niveles de complejidad:

a) Parpadeo de un LED del puerto C independientemente del puerto A.

b) Encender un LED en función de dos interruptores del puerto A.

c) Coche fantástico: Los led rotan de un extremo a otro cambiando de dirección al llegar a éstos, simulando así el efecto de las luces del coche fantástico.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa ensamblador MPLAB y el grabador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

a) Parpadeo de un led del puerto C

Para realizar el parpadeo de un led se debe encender éste y luego apagarlo, cosa que parece muy sencilla, pero cabe tener en cuenta que se debe realizar un retraso entre estas dos acciones puesto que en caso contrario no sería posible visualizar el cambio. Para realizar tal retraso se procede a crear una rutina de espera, la cual es definida al final del programa (Retardo) y es llamada desde éste (CALL Retardo). Para crear este retraso lo único que se intenta es hacer perder el tiempo al microcontrolador haciendo decrementos sucesivos de una o varias variables, pudiendo variar el tiempo de retraso en función del valor asignado. Estos valores pueden ser definidos al principio del programa obteniendo un retraso de valor fijo o antes de la llamada a la subrutina con lo cual conseguiremos un retardo adecuado a cada situación.

b) Control encendido Led

En este segundo programa se pretende controlar el encendido de un led mediante dos pulsadores distintos, es decir, el pulsador conectado a RA0 encenderá el led y el conectado a RA1 lo apagará.

De esta forma conseguimos trabajar con los puertos de entrada, aprendiendo a configurarlos como tales y a utilizar las instrucciones de comprobación del estado del bit.

c) Coche fantástico

Con este tercer y último programa se pretende profundizar al usuario en el listado de instrucciones en ensamblador que se puede encontrar en la hoja de características del microcontrolador. Aunque este listado ya se ha utilizado en los ejemplos anteriores para las acciones de encender y parar o aquellas de chequeo de bit, en este caso deberá escoger una instrucción algo más compleja, en función de la cual, el programa aumentará o disminuirá en dificultad. Por ello se recomienda detenerse unos minutos en este punto para observar más en detalle las opciones que nos ofrecen el conjunto de instrucciones de nuestro microcontrolador, que además, nos serán muy útiles en las siguientes prácticas.

En este caso deberemos diseñar un movimiento de luces que simule el coche fantástico. Por lo tanto, deberá encenderse el led conectado a RC0, y posteriormente la secuencia C0-C1-C2-C3-C4-C5-C6-C7 y al llegar al último led se repetirá el movimiento contrario.

4. SOLUCIONARIO

En esta práctica se introduce el apartado solucionario, en el cual se irán mostrando los programas diseñados para la entrenadora PIC-vBoard que se encuentran comprobados y funcionando al completo. Los siguientes apartados carecerán de introducción puesto que únicamente sirven al profesor como referencia a la posible solución tomada para cada práctica.

```
;*******************************************************************************************
*
;       Programa 2.1: Parpadeo Led
;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
;       Autor: Víctor Bueno Álvez
;
;       Este programa hace parpadear un led conectado al pin 0 del puerto C, su frecuencia
se
;       puede variar mediante las variables de la rutina de retraso.
;       Microcontrolador: PIC16F877A
;       Frecuencia: 4 MHZ
;       Versión 1.0
;*******************************************************************************************
*

;**********Zona de Definiciones**********

        __CONFIG _WDT_OFF&_PWRTE_ON&_HS_OSC&_LVP_OFF&_CP_OFF ; Configuración para el
programador

        LIST    p=16F877A
        INCLUDE <P16F877A.INC>

N       EQU 0x00    ;Variables
cont1   EQU 0x20    ;para subrutina
cont2   EQU 0x21    ;retardo

        ORG 0x00 ; Inicio de programa

;*************Zona de Programa*************

Inicio
    bcf     STATUS,RP0  ; Accede a
    bcf     STATUS,RP1  ; banco 0
    clrf    PORTC       ; Limpia PORTC
    bsf     STATUS,RP0  ; Accede a banco 1
    clrf    TRISC       ; Configura todos las patitas de PORTC como salidas
    bcf     STATUS,RP0  ; Regresa a banco 0
Led
    bsf     PORTC,0     ; La línea RC0 de PORTC toma el valor de 1, se enciende el LED
    call    Retardo     ; Llamada a la rutina de retardo
    bcf     PORTC,0     ; La línea RC0 de PORTC toma el valor de 0, se apaga el LED
    call    Retardo     ; Llamada a la rutina de retardo
    goto    Led         ; Va a la etiqueta Led

;*****Subrutinas

Retardo             ;Rutina de retardo
    movlw   N
    movwf   cont1
Ciclo1
    movlw   N
    movwf   cont2
Ciclo2
    decfsz  cont2,1
    goto    Ciclo2
    decfsz  cont1,1
    goto    Ciclo1
    return          ;Retorno a la llamada de rutina de retardo.

    END ;Fin de programa
```

```
;*****************************************************************************************
*
;       Programa 2.2: Control led 2 pulsadores
;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
;       Autor: Víctor Bueno Álvez
;
;       Este programa enciende o apaga un led conecado en RC0, en función del estado de dos
;       pulsadores conectados en RA0 y RA1.
;       Microcontrolador: PIC16F877A
;       Frecuencia: 4 MHZ
;       Versión: 1.0
;*****************************************************************************************
*

;***************Zona de Definiciones***************

    __CONFIG _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF ; Configuración para el programador

    LIST    p=16F877A
    INCLUDE <P16F877A.INC>

    ORG 0x00        ; Inicio de programa

;*****************Zona de Programa*****************

Inicio
    bcf     STATUS,RP0  ; Accede a banco 0
    bcf     STATUS,RP1
    clrf    PORTC       ; Limpia PORTC
    clrf    PORTA       ; Limpia PORTA
    bsf     STATUS,RP0  ; Accede a banco 1
    clrf    TRISC       ; Configura todos las patitas de PORTC como salidas
    movlw   0x06        ; Configura el puerto A
    movwf   ADCON1      ; como digital
    movlw   0xFF
    movwf   TRISA       ; Configura todos las patitas de PORTA como entradas
    bcf     STATUS,RP0  ; Regresa a banco 0

Compr1
    btfss   PORTA,0     ; Comprueba el estado de RA0
    bcf     PORTC,0     ; Apaga el led si RA0 = 1
    goto    Compr2      ; y salta ha comprovar el de RA1 si es 0
Compr2
    btfss   PORTA,1     ; Comprueba el estado de RA1
    bsf     PORTC,0     ; Enciende el led si RA1 = 1
    goto    Compr1      ; y salta ha comprovar el de RA0 si es 0

    END             ;Fin de programa
```

```
;*****************************************************************************************
*
;       Programa 2.3: Rotación puerto C (Coche fantástico)
;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
;       Autor: Víctor Bueno Álvez
;
;       Este programa enciende secuencialmente los leds conectados al puerto C y cambiando
;       el sentido de rotación al llegar al último led
;       Microcontrolador: PIC16F877A
;       Frecuencia: 4 MHZ
;       Versión: 1.0
;*****************************************************************************************
*


;***************Zona de Definiciones***************

    __CONFIG _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF ; Configuración para el programador

    LIST p=16F877A
    INCLUDE <P16F877A.INC>

    ORG 0x00 ; Inicio de programa

N       EQU 0x00
cont1   EQU 0x20
cont2   EQU 0x21

;*****************Zona de Programa******************

Inicio

    bcf     STATUS,RP0  ; Accede a banco 0
    bcf     STATUS,RP1
    clrf    PORTC       ; Limpia PORTC
    bsf     STATUS,RP0  ; Accede a banco 1
    clrf    TRISC       ; Configura todos las patitas de PORTC como salidas
    bcf     STATUS,RP0  ; Regresa a banco 0

    bsf     PORTC,0         ; La línea RC0 de PORTC toma el valor de 1, se enciende el LED 1

LedSecDer
    call    Retardo     ; Llamada a la rutina de retardo
    rlf     PORTC,1     ; Rota los leds hacia la izquierda
    btfss   PORTC,7     ; Detecta si ha llegado al último led
    goto    LedSecDer   ; Va a la etiqueta LedSec

LedSecIz
    call    Retardo     ; Llamada a la rutina de retardo
    rrf     PORTC,1     ; Rota los leds hacia la derecha
    nop
    decfsz  PORTC,0     ; Detecta si ha llegado al último led
    goto    LedSecIz    ; Va a la etiqueta LedSecIz
    goto    LedSecDer   ; Va a la etiqueta LedSecDer


;***************Subrutinas***************

Retardo             ;Rutina de retardo
    movlw   N
    movwf   cont1
Ciclo1
    movlw   N
    movwf   cont2
Ciclo2
    decfsz  cont2,1
    goto    Ciclo2
    decfsz  cont1,1
    goto    Ciclo1
    return          ;Retorno a la llamada de rutina de retardo.

END ;Fin de programa
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 3 – Contador visualizador 7 segmentos

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En esta práctica se trabajará con un dispositivo de visualización muy utilizado en el pasado y que continúa demostrando su utilidad en algunas aplicaciones debido a sus grandes prestaciones en características tales como gran visibilidad y robusteza, estamos hablando de los visualizadores de 7 segmentos.

Para el trabajo con estos visualizadores se harán dos programas diferentes, el primero en el cual se hará un contador de 0 a 9 utilizando así un solo de estos dispositivos, permitiendo que nos familiarizaremos con el funcionamiento del decodificador BCD/7 segmentos así como con los transistores de conmutación. En el segundo programa pasaremos a realizar un contador de 0 a 9999 utilizando así los 4 visualizadores y una técnica muy conocida en este tipo de visualización, la visualización dinámica. Este método consiste en el encendido consecutivo, de una forma rápida, de cada uno de los dígitos que formen un número, de esta forma nunca se tendrá más de 1 visualizador encendido aunque al usuario le parezca estar viendo los cuatro encendidos a la vez. Este método es muy útil en este tipo de visualizadores debido a su elevado consumo, lo que sería decisivo en aplicaciones en las que utilicemos varios visualizadores como es el caso.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa ensamblador MPLAB y el grabador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

Antes de comenzar con la implementación de esta práctica se definirán las conexiones del controlador 4543 que están realizadas en la placa para facilitar la comprensión del hardware, lo que repercutirá en una mayor facilidad de programación.

Por un lado están las conexiones de control del decodificador las cuales nos permitirán introducir el código correspondiente al número a mostrar y se encuentran conectadas a la parte baja del puerto D del microcontrolador y por otro las conexiones de los transistores de activación de los visualizadores, que se encuentran conectadas en la parte alta. Las conexiones están implementadas como se indica en la siguiente tabla:

Tabla 1. *Conexión visualización 7 segmentos*

Pin	Función
D0	Pin D0 decodificador
D1	Pin D1 decodificador
D2	Pin D2 decodificador
D3	Pin D3 decodificador
D4	Activación millares
D5	Activación centenas
D6	Activación decenas
D7	Activación unidades

Además, se debe tener en cuenta que los dos jumper de selección LCD/7seg. Deben encontrarse en su situación inferior para desactivar el módulo LCD y activar el controlador de 7 segmentos.

A continuación se adjunta una tabla extraída de la hoja de características del controlador 4543 en donde se indica el estado de las diferentes salidas en función del valor de las entradas ya que esta relación puede variar entre un controlador u otro y es necesario para la programación del microcontrolador.

Inputs							Outputs							
$\overline{LE}$	BL	PH [2]	D3	D2	D1	D0	Qa	Qb	Qc	Qd	Qe	Qf	Qg	Display
X	H	L	X	X	X	X	L	L	L	L	L	L	L	blank
H	L	L	L	L	L	L	H	H	H	H	H	H	L	0
H	L	L	L	L	L	H	L	H	H	L	L	L	L	1
H	L	L	L	L	H	L	H	H	L	H	H	L	H	2
H	L	L	L	L	H	H	H	H	H	H	L	L	H	3
H	L	L	L	H	L	L	L	H	H	L	L	H	H	4
H	L	L	L	H	L	H	H	L	H	H	L	H	H	5
H	L	L	L	H	H	L	H	L	H	H	H	H	H	6
H	L	L	L	H	H	H	H	H	H	L	L	L	L	7
H	L	L	H	L	L	L	H	H	H	H	H	H	H	8
H	L	L	H	L	L	H	H	H	H	H	L	H	H	9
H	L	L	H	L	H	X	L	L	L	L	L	L	L	blank
H	L	L	H	H	X	X	L	L	L	L	L	L	L	blank
L	L	L	X	X	X	X	n.c.							n.c
as above		H	as above				inverse of above							as above

Figura 1. *Características controlador 4543*

Una vez definidos los aspectos correspondientes a la conexión del controlador con el microcontrolador y la tabla de correspondencias entre las entradas y el número mostrada se procede a definir los dos programas a realizar en esta práctica.

a) Contador 0-9

En este primer apartado deberemos activar el visualizador correspondiente al digito de las unidades y realizar sobre éste un contador de 0 a 9 implementando el retardo adecuado para la correcta visualización del incremento, por ejemplo 1 segundo por número. El contador deberá ser repetitivo, es decir, una vez haya mostrado el número 9 volverá a comenzar por el 0.

b) Contador 0-9999

En este segundo apartado partiremos del programa realizado en el apartado anterior para conseguir que una vez el dígito de las unidades llegue a 9 éste se reinicialice y el de las decenas incremente en 1 y así sucesivamente hasta el número 9999 donde deberemos inicializar otra vez el contador y reiniciar la cuenta. Además, en este segundo apartado se debe prestar especial atención en los retardos utilizados para conseguir, por ejemplo, un tiempo de incremento de 1 segundo como en el apartado anterior, peor un tiempo de refresco de número menor, para la correcta visualización.

4. SOLUCIONARIO

```
1   ;*****************************************************************************************
    *
2   ;       Programa 3.1: Contador 0-9 7 segmentos
3   ;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
4   ;       Autor: Víctor Bueno Alvez
5   ;
6   ;       Este programa muestra sucesivamente los numeros del 0 al 9 en un display de 7
    segmentos
7   ;       conectado a través de un decodificador al puerto D del microcontrolador.
8   ;       Microcontrolador: PIC16F877A
9   ;       Frecuencia: 4 MHZ
10  ;       Versióm: 1.0
11  ;*****************************************************************************************
    *
12
13  ;*****Zona de Definiciones
14
15      __CONFIG _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF ; Configuración para el programador
16
17      LIST p=16F877A
18      INCLUDE <P16F877A.INC>
19
20      ORG 0x00 ; Inicio de programa
21
22  PDel0   EQU 0x20
23  PDel1   EQU 0x21
24  PDel2   EQU 0x22
25
26  ;*****Zona de Programa
27
28      bcf     STATUS,RP0  ; Accede a banco 0
29      bcf     STATUS,RP1
30      clrf    PORTD       ; Limpia PORTD
31      bsf     STATUS,RP0  ; Accede a banco 1
32      clrf    TRISD       ; Configura todos las patitas de PORTD como salidas
33      bcf     STATUS,RP0  ; Regresa a banco 0
34
35  Ini
36      movlw   b'10000000' ;Activa display unidades y envía un 0
37      movwf   PORTD
38      call    Retardo
39  Cont
40      incf    PORTD,1     ;Incrementa el valor numérico
41      call    Retardo
42      btfss   PORTD,3     ; Comprube asi se ha llegado al nº8
43      goto    Cont        ; Va a la etiqueta Cont
44      btfss   PORTD,0     ; Comprube asi se ha llegado al nº9
45      goto    Cont        ; Va a la etiqueta Cont
46      goto    Ini         ; Va a la etiqueta Ini
47
48  ;***************Zona Subrutinas***************
49
50  Retardo
51          movlw     .14       ; 1 set numero de repeticion  (C)
52          movwf     PDel0     ; 1 |
53  PLoop0
54          movlw     .72       ; 1 set numero de repeticion  (B)
55          movwf     PDel1     ; 1 |
56  PLoop1
57          movlw     .247      ; 1 set numero de repeticion  (A)
58          movwf     PDel2     ; 1 |
59  PLoop2
60          clrwdt              ; 1 clear watchdog
61          decfsz    PDel2, 1  ; 1 + (1) es el tiempo 0  ? (A)
62          goto      PLoop2    ; 2 no, loop
63          decfsz    PDel1,  1 ; 1 + (1) es el tiempo 0  ? (B)
64          goto      PLoop1    ; 2 no, loop
65          decfsz    PDel0,  1 ; 1 + (1) es el tiempo 0  ? (C)
66          goto      PLoop0    ; 2 no, loop
67  PDelL1
68          goto PDelL2         ; 2 ciclos delay
69  PDelL2
70          clrwdt              ; 1 ciclo delay
71          return              ; 2+2 Fin.
72
73  END ;Fin de programa
```

```
;*****************************************************************************************
**
;       Programa 3.2: Contador 0-9999 7 segmentos
;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
;       Autor: Víctor Bueno Alvez
;
;       Este programa muestra sucesivamente los numeros del 0 al 9999 en cuatro displays
;       de 7 segmentos conectado a través de un decodificador al puerto D.
;       Microcontrolador: PIC16F877A
;       Frecuencia: 4 MHZ
;       Versión 1.0
;*****************************************************************************************
**

;***************Zona de Definiciones***************

    __CONFIG _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF ; Configuración para el
programador

    LIST p=16F877A
    INCLUDE <P16F877A.INC>

    ORG 0x00 ; Inicio de programa

PDel0   EQU 0x20
PDel1   EQU 0x21

CBLOCK 0X22
ValorUn, ValorDec, ValorCent, ValorMil, IncRetardo, contador, rep
ENDC

;*****************Zona de Programa*****************

    bcf STATUS,RP0 ; Accede a banco 0
    bcf STATUS,RP1
    clrf PORTD ; Limpia PORTD
    bsf STATUS,RP0 ; Accede a banco 1
    clrf TRISD ; Configura todos las patitas de PORTD como salidas
    bcf STATUS,RP0 ; Regresa a banco 0

    movlw b'10000000'   ;Inicializa a 0 y activa el display
    movwf ValorUn
    movlw b'01000000'   ;Inicializa a 0 y activa el display
    movwf ValorDec
    movlw b'00100000'   ;Inicializa a 0 y activa el display
    movwf ValorCent
    movlw b'00010000'   ;Inicializa a 0 y activa el display
    movwf ValorMil

    movlw 0x00
    movwf contador
    movlw 0x03
    movwf rep

inDisplay
    movlw 0x10
    movwf rep
    incf ValorUn, 1 ; Incrementamos el valor

Display
    movf ValorUn, W ; Muevo ValorUn a W
    movwf PORTD ; ponemos resultado en PORTD (el display)
    call Retardo ; Llamada a la rutina de retardo

    movf ValorDec, W ; Muevo ValorBin a W
    movwf PORTD ; ponemos resultado en PORTD (el display)
    call Retardo ; Llamada a la rutina de retardo

    movf ValorCent, W ; Muevo ValorBin a W
    movwf PORTD ; ponemos resultado en PORTD (el display)
    call Retardo ; Llamada a la rutina de retardo

    movf ValorMil, W ; Muevo ValorBin a W
    movwf PORTD ; ponemos resultado en PORTD (el display)
    call Retardo ; Llamada a la rutina de retardo
```

```
73
74
75      btfsc ValorUn, 3 ;Si se activa el bit 4 de las unidades
76      btfss ValorUn, 1 ;Si se activa el bit 4 de las unidades ;Salta aqui si seactiva bit 4
77      goto cont                                               ;Salta aquisi bit 4 no esta
    activo
78      goto Inc                                                ;Salta aqui sibit 4 y 1 son 1
79
80  cont
81      decfsz  rep,1
82      goto Display ; Va a la etiqueta LedSec
83      goto inDisplay
84
85
86  Inc
87      movlw b'10000000'
88      movwf ValorUn
89      incf ValorDec,1
90      btfsc ValorDec, 3 ;Si se activa el bit 4 de las unidades
91      btfss ValorDec, 1 ;Si se activa el bit 4 de las unidades    ;Salta aqui si seactiva
    bit 4
92      goto Display
93      movlw b'01000000'
94      movwf ValorDec
95      incf ValorCent,1
96      btfsc ValorCent, 3 ;Si se activa el bit 4 de las unidades
97      btfss ValorCent, 1 ;Si se activa el bit 4 de las unidades   ;Salta aqui si seactiva
    bit 4
98      goto Display
99      movlw b'00100000'
100     movwf ValorCent
101     incf ValorMil,1
102     goto Display ; Va a la etiqueta LedSec
103
104 ;*****************Zona de Subrutinas*****************
105
106 Retardo  movlw     .249      ; 1 set numero de repeticion
107         movwf     PDel0     ; 1 |
108 PLoop0  clrwdt              ; 1 clear watchdog
109 PDelL1  goto PDelL2         ; 2 ciclos delay
110 PDelL2  goto PDelL3         ; 2 ciclos delay
111 PDelL3
112         decfsz    PDel0, 1  ; 1 + (1) es el tiempo 0  ?
113         goto      PLoop0    ; 2 no, loop
114 PDelL4  goto PDelL5         ; 2 ciclos delay
115 PDelL5  clrwdt              ; 1 ciclo delay
116         return              ; 2+2 Fin.
117
118
119     END ;Fin de programa
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 4 – Contador visualizador LCD

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En esta práctica se trabajará con otro tipo de visualización que conjuntamente con la estudiada en la práctica anterior mediante visualizadores de 7 segmentos forman dos de los principales métodos utilizados en los circuitos con microcontroladores. Este dispositivo a substituido al antiguo visualizador de 7 segmentos en algunas aplicaciones en donde las propiedades en cuanto a menor consumo o mayor flexibilidad y cantidad de información mostrada del visualizador LCD son necesarias y preferibles antes de la buena visualización y luminosidad de los anteriores de 7 segmentos.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa ensamblador MPLAB y el grabador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

Para comparar los dos tipos de visualización comentados procederemos a realizar el mismo programa que en la práctica anterior, es decir, un contador de 0 a 9999 pero en este caso con el visualizador LCD. De esta forma observaremos la mayor flexibilidad de éste y la gran capacidad de mostrar información que posee, permitiéndonos mostrar además del valor del contador una cabecera delante de éste, por ejemplo: (Contador: 9999).

Lo primero que observamos si miramos alguna hoja del fabricante de estos dispositivos es que disponen de unas rutinas de inicialización y de configuración de los distintos parámetros así como rutinas de envío de carácter, avance de línea, etc. Por ello antes de comenzar a trabajar con un dispositivo de este tipo deberemos crear una librería específica o utilizar una de las muchas disponibles en la web siempre y cuando sean compatibles con nuestra disposición de pines. Además de la disposición de pines, debemos saber que los visualizadores LCD pueden ser conectados mediante un bus de datos de 4 u 8 pines, siendo el primero el escogido para esta tarjeta, y nuestra rutina debe permitir este tipo de conexión. En nuestro caso se ha partido de la estructura de una rutina existente modificándola para nuestro propósito y el conexionado de nuestra placa. Se debe tener en cuenta que en muchas ocasiones y sobre todo cuando se tiene libertad para realizar el conexionado, puede resultar conveniente buscar alguna librería existente de una fuente confiable ahorrando así tiempo de programación y posibles problemas.

De todas formas esta librería es necesario escribirla una única vez, adaptando las conexiones a cada una de nuestras aplicaciones.

A continuación se muestran las conexiones de la entrenadora y la librería para su control:

Tabla 1. *Conexiones LCD*

Pin	Función
D0	Enable
D1	RS
D2	R/W
D4	Data4
D5	Data5
D6	Data6
D7	Data7

```
1    ; REALIZA LA INICIALIZACION DEL LCD. UNA INTERFAZ DE 4 BITS,
2
3    ; 2 LINEAS Y 5X7 PIXELS.
4
5    ;
6
7    ;   RS  ---->   RD1     D7  ---->   RD7
8
9    ;   RW  ---->   RD2     D6  ---->   RD6
10
11   ;   E   ---->   RD0     D5  ---->   RD5
12
13   ;                       D4  ---->   RD4
14
15   ;                       D3  ---->   GND
16
17   ;                       D2  ---->   GND
18
19   ;                       D1  ---->   GND
20
21   ;                       D0  ---->   GND
22
23
24   ; SE DEFINEN PINES DEL PORTA
25
26   #DEFINE D4_LCD  PORTD,4
27   #DEFINE D5_LCD  PORTD,5
28   #DEFINE D6_LCD  PORTD,6
29   #DEFINE D7_LCD  PORTD,7
30   #DEFINE RS  PORTD,1
31   #DEFINE RW  PORTD,2
32   #DEFINE E   PORTD,0
33
34   ; BLOQUE DE COMANDOS
35
36   LCD_LINEA1  EQU 0X80    ;COLOCA EL CURSOR EN LA POSICION 1 LINEA 1
37   LCD_LINEA2  EQU 0XC0    ;COLOCA EL CURSOR EN LA POSICION 1 LINEA 2
38   LCD_BORRAR  EQU 0X01    ;CLR LCD, CURSOR EN LA POSICION 1 LINEA 1
39   LCD_INICIO  EQU 0X02    ;COLOCA EL CURSOR EN LA POSICION 1 LINEA 1
40   LCD_INC     EQU 0X06    ;INC POSICION CURSOR DESPUES DE CADA CARACTER
41   LCD_DEC     EQU 0X04    ;DEC POSICION CURSOR DESPUES DE CADA CARACTER
42   LCD_ON      EQU 0X0C    ;ENCIENDE LCD
43   LCD_OFF     EQU 0X08    ;APAGA LCD
44   LCD_CURSON  EQU 0X0E    ;ENCIENDE LCD Y EL CURSOR (_CURSOR)
45   LCD_CURSOFF EQU 0X0C    ;APAGA EL CURSOR
46   LCD_CURSBLK EQU 0X0D    ;ENCIENDE LCD Y PARPADEA EL CURSOR ([]CURSOR)
47   LCD_IZDA    EQU 0X18    ;DESPLAZA LOS CARACTERES MOSTRADOS A LA IZQUIERDA
48   LCD_DECHA   EQU 0X1C    ;DESPLAZA LOS CARACTERES MOSTRADOS A LA DERECHA
49   LCD_CURSIZ  EQU 0X10    ;MUEVE EL CURSOR UNA POSICION A LA IZQUIERDA
50   LCD_CURSDE  EQU 0X14    ;MUEVE EL CURSOR UNA POSICION A LA DERECHA
51   LCD_4BITS   EQU 0X28    ;INTERFACE 4 BITS, 2 LINEAS, 5X7 PIXELES
52   LCD_CGRAM   EQU 0X40    ;GENERADOR DE CARACTERES DE USUARIO RAM
53   PDel0       EQU 0x20
54   PDel1       EQU 0x21
55
56   ; RUTINA PARA LA CONFIGURACION DEL LCD
57
58   LCD_CONFIG
59
60        CALL    RETARDO_150ms
61        MOVLW   B'00001111'
62        ANDWF   PORTD,F     ;BORRA PORTA_HIGH RESPETANDO LAS RESTANTES
63        MOVLW   0X30
64        IORWF   PORTD,F     ;INTERFACE DE 8 BITS
65        BCF RW      ;MODO ESCRITURA
66        BCF RS      ;MODO COMANDO
67        CALL    LCD_E
68        CALL    RETARDO_150ms       ;RETARDO SUGERIDO POR EL FABRICANTE
69        CALL    RETARDO_150ms;**************************************************
70
71        CALL    LCD_E
72        CALL    RETARDO_150ms       ;RETARDO SUGERIDO POR EL FABRICANTE
73        CALL    RETARDO_150ms;**************************************************
74
75        CALL    LCD_E
```

```
76        CALL    RETARDO_150ms       ;RETARDO SUGERIDO POR EL FABRICANTE
77        CALL    RETARDO_150ms;**************************************************
78
79        MOVLW   B'00001111'
80        ANDWF   PORTD,F     ;BORRA PORTA_HIGH RESPETANDO LAS RESTANTES
81        MOVLW   0X20
82
83        IORWF   PORTD,F     ;INTERFACE DE 4 BITS
84        CALL    LCD_E
85
86        MOVLW   LCD_4BITS       ;INTERFACE DE 4 BITS, 2 LINEAS Y 5X7 PIXELS
87        CALL    LCD_CMD
88        MOVLW   LCD_ON
89        CALL    LCD_CMD
90        MOVLW   LCD_BORRAR
91        CALL    LCD_CMD
92
93        RETURN
94
95
96    ; RUTINA QUE ENTREGA UN PULSO A ENABLE DEL LCD
97
98    LCD_E
99
100       BSF E
101       NOP
102       NOP
103
104       BCF E
105       RETURN
106
107   ; RUTINA QUE VERIFICA EL ESTADO DEL FLAG BUSY DEL LCD.
108
109   LCD_BUSY
110
111       ;BANK_1
112       BCF STATUS,RP1
113       BSF STATUS,RP0
114       MOVLW   B'11110000'
115       IORWF   TRISD,F     ;PORTD_HIGH COMO ENTRADA
116
117       ;BANK_0
118       BCF STATUS,RP0
119       BSF RW      ;MODO LECTURA
120       BCF RS      ;MODO COMANDO
121       NOP
122
123   LCD_BUSY3
124
125       BSF E
126       NOP
127       NOP
128
129       MOVFW   PORTD
130       MOVWF   LCD_TEMP
131       NOP
132       NOP
133
134       BCF E
135       NOP
136       NOP
137
138       BSF E
139       NOP
140       NOP
141       BCF E
142       BTFSC   LCD_TEMP,3
143       GOTO    LCD_BUSY3
144
145       ;BANK_1
146       BCF STATUS,RP1
147       BSF STATUS,RP0
148       MOVLW   B'00001111'
149       ANDWF   TRISD,F     ;PORTA_HIGH COMO SALIDA
150       ;BANK_0
```

```
151      BCF STATUS,RP0
152
153      RETURN
154
155  ; RUTINA QUE ENVIA UN COMANDO AL LCD
156
157  LCD_CMD
158
159      MOVWF   LCD_CHAR        ;SALVA COMANDO PARA SER ENVIADO
160      CALL    LCD_BUSY
161      BCF RW      ;MODO ESCRITURA
162      BCF RS      ;MODO COMANDO
163      GOTO    $+.5
164
165  ; RUTINA QUE VISUALIZA EN EL LCD EL CONTENIDO CARGADO EN W (W=ASCII)
166
167  LCD_DATA
168
169      MOVWF   LCD_CHAR        ;SALVA COMANDO PARA SER ENVIADO
170      CALL    LCD_BUSY
171      BCF RW      ;MODO ESCRITURA
172      BSF RS      ;MODO CARACTER
173      BCF D4_LCD
174      BCF D5_LCD
175      BCF D6_LCD
176      BCF D7_LCD
177
178  ; SEND NYBLE MSB
179
180      BTFSC   LCD_CHAR,7
181      BSF D7_LCD
182      BTFSC   LCD_CHAR,6
183      BSF D6_LCD
184      BTFSC   LCD_CHAR,5
185      BSF D5_LCD
186      BTFSC   LCD_CHAR,4
187      BSF D4_LCD
188      CALL    LCD_E
189
190  ; SEND NYBLE LSB
191
192      BCF D4_LCD
193      BCF D5_LCD
194      BCF D6_LCD
195      BCF D7_LCD
196      BTFSC   LCD_CHAR,3
197      BSF D7_LCD
198      BTFSC   LCD_CHAR,2
199      BSF D6_LCD
200      BTFSC   LCD_CHAR,1
201      BSF D5_LCD
202      BTFSC   LCD_CHAR,0
203      BSF D4_LCD
204      CALL    LCD_E
205
206      RETURN
207
208
209  ;-------------------------------------------------------------
210  RETARDO_5ms
211          movlw     .6        ; 1 set numero de repeticion  (B)
212          movwf     PDel0     ; 1 |
213  PLoop1  movlw     .207      ; 1 set numero de repeticion  (A)
214          movwf     PDel1     ; 1 |
215  PLoop2  clrwdt              ; 1 clear watchdog
216          decfsz    PDel1, 1  ; 1 + (1) es el tiempo 0  ? (A)
217          goto      PLoop2    ; 2 no, loop
218          decfsz    PDel0,  1 ; 1 + (1) es el tiempo 0  ? (B)
219          goto      PLoop1    ; 2 no, loop
220  PDelL1  goto PDelL2         ; 2 ciclos delay
221  PDelL2  clrwdt              ; 1 ciclo delay
222          return              ; 2+2 Fin.
223
224  ;-------------------------------------------------------------
225  RETARDO_150ms
```

```
226          movlw     .117      ; 1 set numero de repeticion  (B)
227          movwf     PDel0     ; 1 |
228  PLoop3  movlw     .213      ; 1 set numero de repeticion  (A)
229          movwf     PDel1     ; 1 |
230  PLoop4  clrwdt               ; 1 clear watchdog
231  PDelL3  goto      PDelL4         ; 2 ciclos delay
232  PDelL4
233          decfsz    PDel1, 1  ; 1 + (1) es el tiempo 0  ? (A)
234          goto      PLoop4    ; 2 no, loop
235          decfsz    PDel0,  1 ; 1 + (1) es el tiempo 0  ? (B)
236          goto      PLoop3    ; 2 no, loop
237          clrwdt               ; 1 ciclo delay
238          return               ; 2+2 Fin.
```

4. SOLUCIONARIO

```
;*******************************************************************************************
**
;       Programa 4.1: Contador 0-9999 LCD
;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
;       Autor: Víctor Bueno Alvez
;
;       Este programa muestra sucesivamente los numeros del 0 al 9999 en un display LCD
;       conectado al puerto D del microcontrolador.
;       Microcontrolador: PIC16F877A
;       Frecuencia: 4 MHZ
;       Versión 1.0
;*******************************************************************************************
**

;***************Zona de Definiciones***************

    __CONFIG   _CP_OFF &  _WDT_OFF & _PWRTE_ON & _HS_OSC & _LVP_OFF
    LIST       P=16F877A
    INCLUDE  <P16F877A.INC>

LCD_TEMP EQU 0x25   ; Definiciones necesarias
LCD_CHAR EQU 0x27   ; al utilizar la librería del LCD

CBLOCK 0x30         ; Variables para los valores de los contadores
REG1
REG2
RLSD
RMSD
ENDC

;***************Zona de Programa***************

    ORG 0

Inicio
    bcf     STATUS,RP0  ; Accede a
    bcf     STATUS,RP1  ; banco 0
    clrf    PORTD       ; Limpia PORTD
    bsf     STATUS,RP0  ; Accede a banco 1
    clrf    TRISD       ; Configura todos las patitas de PORTD como salidas
    bcf     STATUS,RP0  ; Regresa a banco 0

    call    LCD_CONFIG
    call    LCD_DATA

    call    RETARDO_150ms

    movlw   0x00
    movwf   REG1
    movlw   0x00
    movwf   REG2


    movlw   LCD_INICIO  ; Inicializa el LCD
    call    LCD_CMD     ; en la primera línea

    movlw   'P'         ; Envía letra a letra
    call    LCD_DATA    ; las palabras
    movlw   'r'         ; ProgramaContador
    call    LCD_DATA
    movlw   'o'
    call    LCD_DATA
    movlw   'g'
    call    LCD_DATA
    movlw   'r'
    call    LCD_DATA
    movlw   'a'
    call    LCD_DATA
    movlw   'm'
    call    LCD_DATA
    movlw   'a'
    call    LCD_DATA
    movlw   'C'
    call    LCD_DATA
    movlw   'o'
```

```
74        call    LCD_DATA
75        movlw   'n'
76        call    LCD_DATA
77        movlw   't'
78        call    LCD_DATA
79        movlw   'a'
80        call    LCD_DATA
81        movlw   'd'
82        call    LCD_DATA
83        movlw   'o'
84        call    LCD_DATA
85        movlw   'r'
86        call    LCD_DATA
87
88    visualizar
89
90        movlw   LCD_LINEA2  ; Inicializa el LCD
91        call    LCD_CMD     ; en la segunda linea
92        movlw   'C'         ; Envía la palabra
93        call    LCD_DATA    ; Contador:
94        movlw   'o'
95        call    LCD_DATA
96        movlw   'n'
97        call    LCD_DATA
98        movlw   't'
99        call    LCD_DATA
100       movlw   'a'
101       call    LCD_DATA
102       movlw   'd'
103       call    LCD_DATA
104       movlw   'o'
105       call    LCD_DATA
106       movlw   'r'
107       call    LCD_DATA
108       movlw   ':'
109       call    LCD_DATA
110       movlw   ' '
111       call    LCD_DATA
112
113       movf    REG2,W
114       call    BINconvBCD
115       movf    RMSD,W
116       addlw   0X30
117       call    LCD_DATA
118       movlw   '.'
119       call    LCD_DATA
120       movf    RLSD,W
121       addlw   0X30
122       call    LCD_DATA
123       movf    REG1,W
124       call    BINconvBCD
125       movf    RMSD,W
126       addlw   0X30
127       call    LCD_DATA
128       movf    RLSD,W
129       addlw   0X30
130       call    LCD_DATA
131
132       call    Incrementar
133
134       call    RETARDO_150ms
135       call    RETARDO_150ms
136
137       goto visualizar
138
139   ;***************Zona de Subrutinas***************
140
141   Incrementar             ; Subrutina para incrementar
142       incf    REG1,F      ; el valor de los registros
143       movf    REG1,W
144       sublw   0X64
145       btfss   STATUS,Z
146       return
147       clrf    REG1
148       incf    REG2,F
```

```
149      movf    REG2,W
150      sublw   0X64
151      btfss   STATUS,Z
152      return
153      clrf     REG2
154      return
155
156
157  BINconvBCD              ; Subrutina para convertir
158      clrf    RMSD        ; los valores binarios a BCD
159      movwf   RLSD        ; para que puedan ser enviados al LCD
160  TENSUB
161      movlw   .10
162      subwf   RLSD,W
163      btfss   STATUS,C
164      return
165      movwf   RLSD
166      incf    RMSD,F
167      goto    TENSUB
168
169
170      INCLUDE  <LCD_int.inc>  ; Subrutinas de control del módulo LCD.
171      END                     ; Fin del programa.
172
173
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 5 – Llave electrónica

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En esta práctica se introducirá al usuario al control de un periférico ampliamente utilizado en aplicaciones que necesiten de una interfaz hombre - máquina como pueden ser aquellas en donde se necesite disponer de un método de entrada de datos alfanuméricos o símbolos de una manera sencilla y con un bajo consumo de pines del microcontrolador, estamos hablando del teclado matricial.

Existe una gran variedad de este tipo de teclados en función del tamaño de su matriz, en nuestro caso se ha utilizado un teclado matricial 4x4 con lo cual disponemos de 16 teclas con los números del 0 al 9 las letras de la A a la D y lo símbolos (* y #).

Para introducir al lector en este tipo de periférico trabajaremos con un programa que utilizará el teclado matricial para realizar un control de acceso por código.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa ensamblador MPLAB y el grabador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

En esta práctica se dispondrá del teclado matricial como elemento de entrada de datos, el visualizador LCD como elemento de salida de datos y un relé como elemento de control, el cual nos abrirá nuestra puerta en caso de que el código sea introducido correctamente.

Con estos elementos deberemos proceder a la introducción de un código secreto, esta tarea la podemos realizar primero por software, simplificando así el programa, y después diseñar un método para poder hacerlo desde el mismo teclado. Una vez volcado nuestro código secreto en el microcontrolador introduciremos los datos desde el teclado y el microcontrolador deberá comprobar si el dato de entrada es igual al valor guardado y mostrar en el LCD si el código es correcto o no, activando el relé en caso afirmativo.

4. SOLUCIONARIO

```
;*****************************************************************************************
;       Programa 5.1: Llave electronica
;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
;       Autor: Víctor Bueno Álvez
;
;       Este programa implementa una llave electrónica en donde programamos un código que luego
;       deberemos introducir por teclado para abrir la puerta.
;       Microcontrolador: PIC16F877A
;       Frecuencia: 4 MHZ
;       Versión 1.0
;*****************************************************************************************

;***************Zona de Definiciones***************

    __CONFIG _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF ; Configuración para el programador

    LIST p=16F877A
    INCLUDE <P16F877A.INC>

LCD_TEMP EQU 0x25
LCD_CHAR EQU 0x27
cont    EQU 0x22
N       EQU 0x00
cont1   EQU 0x20    ;Variables para
cont2   EQU 0x21    ;rutina restardo
num1    EQU 0x24
num2    EQU 0x26
num3    EQU 0x28
CBLOCK  0x30
num4
Tecla
col
rep
PDel2
PDel3
PDel4
ENDC
correcto    EQU 0x34

    ORG 0x00 ; Inicio de programa
    GOTO Inicio

;***************Zona de Programa***************

Inicio

    bcf     STATUS,RP0  ; Accede a
    bcf     STATUS,RP1  ;banco 0
    clrf    PORTB       ; Limpia PORTB
    clrf    PORTC       ; Limpia PORTC
    clrf    PORTD       ; Limpia PORTD
    clrf    PORTE       ; Limpia PORTE
    bsf     STATUS,RP0  ; Accede a banco 1
    movlw   0xF0        ; PortB0-3 Salidas
    movwf   TRISB       ; PortB4-7 Entradas
    bcf     OPTION_REG,7 ; Habilita Resistencias de
                         ; polarización en entradas
    clrf    TRISC       ; Configura todos las patitas de PORTC como salidas
    clrf    TRISD       ; Configura todos las patitas de PORTD como salidas
    clrf    TRISE       ; Configura todos las patitas de PORTE como salidas
    bcf     STATUS,RP0  ; Regresa a banco 0

    call    LCD_CONFIG  ;Inicializamos el LCD
    call    LCD_DATA
    call    RETARDO_150ms

ini
    ;Indicamos el código secreto formado por 4 numeros
    movlw '0'
    movwf num1
    movlw 'A'
```

```
72        movwf num2
73        movlw '1'
74        movwf num3
75        movlw 'B'
76        movwf num4
77
78
79        movlw 0x05
80        movwf rep
81        movlw 0x00
82        movwf correcto
83
84        movlw   LCD_INICIO  ; Inicializa el LCD
85        call    LCD_CMD     ; en la primera línea
86
87        movlw   'C'
88        call    LCD_DATA
89        movlw   'L'
90        call    LCD_DATA
91        movlw   'A'
92        call    LCD_DATA
93        movlw   'V'
94        call    LCD_DATA
95        movlw   'E'
96        call    LCD_DATA
97        movlw   ' '
98        call    LCD_DATA
99        movlw   '4'
100       call    LCD_DATA
101       movlw   ' '
102       call    LCD_DATA
103       movlw   'D'
104       call    LCD_DATA
105       movlw   'I'
106       call    LCD_DATA
107       movlw   'G'
108       call    LCD_DATA
109       movlw   'I'
110       call    LCD_DATA
111       movlw   'T'
112       call    LCD_DATA
113       movlw   'O'
114       call    LCD_DATA
115       movlw   'S'
116       call    LCD_DATA
117       call    RETARDO_1s
118
119       movlw   LCD_BORRAR  ; Inicializa el LCD
120       call    LCD_CMD     ; en la primera línea
121
122   KB_Scan                 ;Escanea el teclado
123       clrf    Tecla           ;Borra Tecla y
124       incf    Tecla,1         ;prepara Tecla para primer c¢digo.
125       movlw   0x0E             ;Saca 0 a la primera fila
126       movwf   PORTB           ;de la Puerta B
127       nop                     ;Nada para estabilizaci¢n de señal.
128
129   Cheq_Col
130       btfss   PORTB,4         ;Primera columna = 0
131       goto    antirebotes     ;Sale si se ha pulsado tecla.
132       incf    Tecla,1         ;Si no tecla pulsada, incrementa tecla.
133       btfss   PORTB,5         ;Segunda columna = 0
134       goto    antirebotes     ;Sale si se ha pulsado tecla.
135       incf    Tecla,1         ;Si no tecla pulsada, incrementa tecla.
136       btfss   PORTB,6         ;Tercera columna = 0
137       goto    antirebotes     ;Sale si se ha pulsado tecla.
138       incf    Tecla,1         ;Si no tecla pulsada, incrementa tecla.
139       btfss   PORTB,7         ;Cuarta columna = 0
140       goto    antirebotes     ;Sale si se ha pulsado tecla.
141       incf    Tecla,1         ;Si no tecla pulsada,incrementa Tecla.
142
143   Ultima_Tecla
144       movlw   d'17'           ;Carga W con el n£mero de Teclas + 1.
145       subwf   Tecla,w         ;y lo compara con el valor actual de Tecla.
146       btfsc   STATUS,Z        ;Si Tecla + 1 = valor actual.
```

```
147      goto    NTeclas         ;No ha sido pulsada ninguna tecla.
148      bsf     STATUS,C        ;Pone a 1 Bit C.
149      rlf     PORTB,f         ;as¡ la Fila 1 pasa a 1 con la rotaci¢n a izqda.
150      goto    Cheq_Col
151
152  NTeclas
153      clrf    Tecla           ;Coloca variable Tecla a 0
154      goto    KB_Scan
155
156  antirebotes ;ahora se espera a que la tecla sea soltada para evitar rebotes
157
158  Espera1
159      btfss   PORTB,4         ;Si no se suelta la tecla FILA 1
160      goto    Espera1         ;vuelve a esperar.
161
162  Espera2
163      btfss   PORTB,5         ;Si no se suelta la tecla FILA 2
164      goto    Espera2         ;vuelve a esperar.
165
166  Espera3
167      btfss   PORTB,6         ;Si no se suelta la tecla FILA 3
168      goto    Espera3         ;vuelve a esperar.
169
170  Espera4
171      btfss   PORTB,7         ;Si no se suelta la tecla FILA 4
172      goto    Espera4         ;vuelve a esperar.
173
174      movlw   '*'
175      call    LCD_DATA
176      call    Retardo
177      movf    Tecla,w         ;pone en w el numero contenido en la variable
178      call    T_Conv          ;llama a la tabla de conversion y retorna
179
180
181
182      decfsz  rep,1
183      goto    compr
184      btfss   correcto,1
185      goto    clavecorrecta
186      goto    claveincorrecta
187
188  compr
189      btfss   correcto,2
190      goto    numero1
191      btfss   correcto,3
192      goto    numero2
193      btfss   correcto,4
194      goto    numero3
195      btfss   correcto,5
196      goto    numero4
197
198
199  numero1
200      subwf   num1,W
201      bsf     correcto,2
202      goto    compr2
203
204  numero2
205      subwf   num2,W
206      bsf     correcto,3
207      goto    compr2
208
209  numero3
210      subwf   num3,W
211      bsf     correcto,4
212      goto    compr2
213
214  numero4
215      subwf   num4,W
216      bsf     correcto,5
217      goto    compr2
218
219  compr2
220      btfss   STATUS,Z
221  Incorrecta
```

```
222      bsf     correcto,1  ;Detecta que és incorrecta
223      goto    KB_Scan
224  Correcta
225
226  clavecorrecta
227      movlw   LCD_INICIO  ; Inicializa el LCD
228      call    LCD_CMD     ; en la primera línea
229      movlw   'C'
230      call    LCD_DATA
231      movlw   'L'
232      call    LCD_DATA
233      movlw   'A'
234      call    LCD_DATA
235      movlw   'V'
236      call    LCD_DATA
237      movlw   'E'
238      call    LCD_DATA
239      movlw   ' '
240      call    LCD_DATA
241      movlw   'C'
242      call    LCD_DATA
243      movlw   'O'
244      call    LCD_DATA
245      movlw   'R'
246      call    LCD_DATA
247      movlw   'R'
248      call    LCD_DATA
249      movlw   'E'
250      call    LCD_DATA
251      movlw   'C'
252      call    LCD_DATA
253      movlw   'T'
254      call    LCD_DATA
255      movlw   'A'
256      call    LCD_DATA
257      call    RETARDO_1s
258
259      bsf     PORTE,2
260
261      call    RETARDO_1s
262
263      bcf     PORTE,2
264      goto    ini
265
266  claveincorrecta
267      movlw   LCD_INICIO  ; Inicializa el LCD
268      call    LCD_CMD     ; en la primera línea
269      movlw   'C'
270      call    LCD_DATA
271      movlw   'L'
272      call    LCD_DATA
273      movlw   'A'
274      call    LCD_DATA
275      movlw   'V'
276      call    LCD_DATA
277      movlw   'E'
278      call    LCD_DATA
279      movlw   ' '
280      call    LCD_DATA
281      movlw   'I'
282      call    LCD_DATA
283      movlw   'N'
284      call    LCD_DATA
285      movlw   'C'
286      call    LCD_DATA
287      movlw   'O'
288      call    LCD_DATA
289      movlw   'R'
290      call    LCD_DATA
291      movlw   'R'
292      call    LCD_DATA
293      movlw   'E'
294      call    LCD_DATA
295      movlw   'C'
296      call    LCD_DATA
```

```
297      movlw   'T'
298      call    LCD_DATA
299      movlw   'A'
300      call    LCD_DATA
301
302      call    RETARDO_1s
303
304      bsf     PORTC,2
305
306      call    RETARDO_1s
307
308      bcf     PORTC,2
309      goto    ini
310
311  T_Conv
312      addwf   PCL,1
313      retlw   '0'             ;Tecla n°0 = 0
314      retlw   '1'             ;Tecla n°1 = 1
315      retlw   '2'             ;Tecla n°2 = 4
316      retlw   '3'             ;Tecla n°3 = 7
317      retlw   'A'             ;Tecla n°4 = A
318      retlw   '4'             ;Tecla n°5 = 2
319      retlw   '5'             ;Tecla n°6 = 5
320      retlw   '6'             ;Tecla n°7 = 8
321      retlw   'B'             ;Tecla n°8 = 0
322      retlw   '7'             ;Tecla n°9 = 3
323      retlw   '8'             ;Tecla n°10 = 6
324      retlw   '9'             ;Tecla n°11 = 9
325      retlw   'C'             ;Tecla n°12 = B
326      retlw   '*'             ;Tecla n°13 = F
327      retlw   '0'             ;Tecla n°14 = E
328      retlw   '#'             ;Tecla n°15 = D
329      retlw   'D'             ;Tecla n°16 = C
330      return
331
332
333  ;***************Zona de Subrutinas***************
334
335  Retardo                 ;Rutina de retardo
336      movlw   N
337      movwf   cont1
338  Rep1
339      movlw   N
340      movwf   cont2
341  Rep2
342      decfsz  cont2,1
343      goto    Rep2
344      decfsz  cont1,1
345      goto    Rep1
346      return              ;Retorno a la llamada de rutina de retardo.
347
348  RETARDO_1s              ;Rutina de retardo de 1 segundo
349      movlw     .14       ; 1 set numero de repeticion  (C)
350      movwf     PDel2     ; 1 |
351  PLoop7
352      movlw     .72       ; 1 set numero de repeticion  (B)
353      movwf     PDel3     ; 1 |
354  PLoop5
355      movlw     .247      ; 1 set numero de repeticion  (A)
356      movwf     PDel4     ; 1 |
357  PLoop6
358      clrwdt              ; 1 clear watchdog
359      decfsz    PDel4, 1  ; 1 + (1) es el tiempo 0  ? (A)
360      goto      PLoop6    ; 2 no, loop
361      decfsz    PDel3,  1 ; 1 + (1) es el tiempo 0  ? (B)
362      goto      PLoop5    ; 2 no, loop
363      decfsz    PDel2,  1 ; 1 + (1) es el tiempo 0  ? (C)
364      goto      PLoop7    ; 2 no, loop
365  PDelL6
366      goto PDelL5         ; 2 ciclos delay
367  PDelL5
368      clrwdt              ; 1 ciclo delay
369      return              ; 2+2 Fin.
370
371
```

```
372       INCLUDE  <LCD_int.inc>  ; Subrutinas de control del módulo
     LCD.;*****************************
373       END ;Fin de programa
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 6 – Control ascensor

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En esta práctica continuaremos trabajando con el teclado matricial, pero en este caso se introducirá una herramienta del microcontrolador muy útil y ampliamente utilizada en el trabajo con este tipo de periféricos, las interrupciones. El microcontrolador dispone de distintas fuentes de interrupciones, desde pines seleccionables específicamente para tal fin, hasta interrupciones en los métodos de comunicación como RS232. En este caso se utilizará una interrupción muy útil que no implica utilizar ningún pin extra y que consiste en la detección de un cambio en el estado de cualquier pin de la parte baja del puerto B.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa ensamblador MPLAB y el grabador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

Para trabajar con las interrupciones se diseñará un sencillo programa que simulará el funcionamiento de un ascensor. Siendo las letras los pulsadores situados en las distintas plantas. De esta forma utilizando la barra de leds simularemos el movimiento del ascensor, el cual también puede ser visualizado en el LCD. Esta práctica se ha diseñado de una forma sencilla para permitir al usuario experimentar libremente con el uso de las interrupciones así como la posible ampliación de la práctica, activación relé simulando abertura puertas, simulación botonera interior ascensor, etc.

4. SOLUCIONARIO

```
;*******************************************************************************************
**
;       Programa 6.1: Teclado interrupciones (Simulación ascensor)
;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
;       Autor: Víctor Bueno Álvez
;
;       Este programa simula el funcionamiento de un ascensor en el cual las teclas ABCD
són los
;       pulsadores situados en las 4 plantas y en la barra de leds se muestra el estado
del ascensor.
;       Microcontrolador: PIC16F877A
;       Frecuencia: 4 MHZ
;       Versión 1.0
;*******************************************************************************************
**

;***************Zona de Definiciones***************
     __CONFIG _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF ; Configuración para el
programador

     LIST p=16F877A
     INCLUDE <P16F877A.INC>

LCD_TEMP    EQU 0x25    ; Definiciones necesarias
LCD_CHAR    EQU 0x27    ; al utilizar la librería del LCD
N           EQU 0x00
cont1       EQU 0x20    ;Variables para
cont2       EQU 0x21    ;rutina restardo
tecla       EQU h'30'
col         EQU h'31'
fila        EQU 0x23

     ORG     0x00 ; Inicio de programa
     GOTO    Inicio

     ORG     0x04    ;Vector Interrupción
     goto    IntRB

;***************Zona de Programa***************

Inicio

     bcf     STATUS,RP0  ; Accede a
     bcf     STATUS,RP1  ;banco 0
     clrf    PORTB ; Limpia PORTB
     clrf    PORTC ; Limpia PORTC
     clrf    PORTD ; Limpia PORTD
     bsf     STATUS,RP0 ; Accede a banco 1
     movlw   0xF0            ; PortB0-3 Salidas
     movwf    TRISB          ; PortB4-7 Entradas
     bcf     OPTION_REG,7    ; Habilita Resistencias de
                             ; polarización en entradas
     clrf    TRISC ; Configura todos las patitas de PORTC como salidas
     clrf    TRISD ; Configura todos las patitas de PORTD como salidas
     bcf     STATUS,RP0 ; Regresa a banco 0

Conf_Int
     bsf     INTCON,RBIE ;Habilitar Interrupción RBIE
     bcf     INTCON,RBIF ;Limpiar bandera de la interrupción
     bsf     INTCON,GIE  ;Habilitar interrupciones

     call    LCD_CONFIG
     call    LCD_DATA

     call    RETARDO_150ms


Espera

     movlw   LCD_INICIO  ; Inicializa el LCD
     call    LCD_CMD     ; en la primera línea

     movlw   'P'         ; Envía letra a letra
     call    LCD_DATA    ; las palabras
     movlw   'L'         ; Envía letra a letra
```

```
71        call    LCD_DATA    ; las palabras
72        movlw   'A'         ; Envía letra a letra
73        call    LCD_DATA    ; las palabras
74        movlw   'N'         ; Envía letra a letra
75        call    LCD_DATA    ; las palabras
76        movlw   'T'         ; Envía letra a letra
77        call    LCD_DATA    ; las palabras
78        movlw   'A'         ; Envía letra a letra
79        call    LCD_DATA    ; las palabras
80        movlw   ' '         ; Envía letra a letra
81        call    LCD_DATA    ; las palabras
82        btfsc   PORTC,0
83        movlw   '0'         ; Envía letra a letra
84        btfsc   PORTC,3
85        movlw   '1'         ; Envía letra a letra
86        btfsc   PORTC,5
87        movlw   '2'         ; Envía letra a letra
88        btfsc   PORTC,7
89        movlw   '3'         ; Envía letra a letra
90        call    LCD_DATA    ; las palabras
91
92        goto    Espera
93
94    ;***************Zona de Interrupción***************
95
96    IntRB
97        bcf     INTCON,GIE  ;Deshabilita las interrupciones
98
99    Detec
100       bcf     PORTB,0
101       btfss   PORTB,7 ;Letra A
102       goto    Rebotes
103
104       btfss   PORTB,6 ;Numero 3
105       goto    Rebotes
106
107       btfss   PORTB,5 ;Numero 2
108       goto    Rebotes
109
110       btfss   PORTB,4 ;Numero 1
111       goto    Rebotes
112
113       goto    FinInt
114
115   Rebotes
116       call    Retardo
117       btfss   PORTB,7 ;Detecta pulsación planta A (cuarta)
118       goto    cuarta
119       btfss   PORTB,6 ;Detecta pulsación planta 3
120       goto    tercera
121       btfss   PORTB,5 ;Detecta pulsación planta 2
122       goto    segunda
123       btfss   PORTB,4 ;Detecta pulsación planta 1
124       goto    primera
125
126       goto    FinInt
127
128   cuarta  btfsc   PORTC,7
129           goto    FinInt
130           rlf     PORTC,1
131           call    Retardo
132           btfss   PORTC,7
133           goto    cuarta
134           clrf    PORTC
135           bsf     PORTC,7
136           goto    FinInt
137
138   tercera btfsc   PORTC,5
139           goto    FinInt
140
141           btfss   PORTC,7
142           Goto    Subir3
143           Goto    Bajar3
144   Subir3  rlf     PORTC,1
145           call    Retardo
```

```
146          btfss   PORTC,5
147          Goto    Subir3
148          goto    FinInt
149  Bajar3  rrf     PORTC,1
150          call    Retardo
151          btfss   PORTC,5
152          goto    Bajar3
153          goto    FinInt
154
155  segunda btfsc   PORTC,3
156          goto    FinInt
157
158          btfss   PORTC,7
159          btfsc   PORTC,5
160          goto    Bajar2
161          goto    Subir2
162  Subir2  rlf     PORTC,1
163          call    Retardo
164          btfss   PORTC,3
165          goto    Subir2
166          goto    FinInt
167  Bajar2  rrf     PORTC,1
168          call    Retardo
169          btfss   PORTC,3
170          goto    Bajar2
171          goto    FinInt
172
173  primera btfsc   PORTC,0
174          goto    FinInt
175
176          rrf     PORTC,1
177          call    Retardo
178          btfss   PORTC,0
179          goto    primera
180          clrf    PORTC
181          bsf     PORTC,0
182
183
184  FinInt
185      bcf     INTCON,RBIF ;Limpiar bandera de la interrupción
186      bsf     INTCON,GIE  ;Habilita las interrupciones
187      retfie
188
189  ;***************Zona de Subrutinas***************
190
191  Retardo ;Rutina de retardo
192      movlw   N
193      movwf   cont1
194  Rep1
195      movlw   N
196      movwf   cont2
197  Rep2
198      decfsz  cont2,1
199      goto    Rep2
200      decfsz  cont1,1
201      goto    Rep1
202      return ;Retorno a la llamada de rutina de retardo.
203
204      INCLUDE  <LCD_int.inc>  ; Subrutinas de control del módulo LCD.
205      END ;Fin de programa
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 7 – Sensado temperatura

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En esta práctica se trabajará con un dispositivo muy utilizado en la vida real y que en este caso se encuentra integrado en el mismo microcontrolador, el convertidor ADC. Este convertidor es prácticamente imprescindible en el mundo digital, ya que muchas operaciones se basan en la medida de una variable analógica y su posterior procesado digital, para finalmente actuar de una manera digital o analógica en el medio. De esta forma este tipo de convertidores se convierten en una herramienta indispensable para el trabajo con microcontroladores.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa ensamblador MPLAB y el grabador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

Para trabajar con el módulo analógico/digital del microcontrolador disponemos de 4 generadores analógicos integrados en la placa además de la posibilidad de conectar otros externos. En esta práctica se utilizará el medidor de temperatura LM35 para así mostrar esta variable por el módulo LCD.

Debemos tener en cuenta que una vez convertido el valor entregado por el ADC a tensión deberemos utilizar la relación de 10 mV/ºC que nos entrega el LM35 para mostrar la temperatura medida. Como ampliación a la práctica se puede mostrar la temperatura de forma visual mediante la barra de leds o utilizando también los visualizadores de 7 segmentos.

4. SOLUCIONARIO

```
;*******************************************************************************************
;       Programa 7.1: Medida temperatura LM35 LCD
;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
;       Autor: Víctor Bueno Álvez
;
;       Este programa muestra el valor de la temperaura adquirido a través de un sensor LM35
;       conectado al canal AN0/RA0 ylo visualiza en un display LCD.
;       Microcontrolador: PIC16F877A
;       Frecuencia: 4 MHZ
;       Versión 1.0
;*******************************************************************************************

;***************Zona de Definiciones***************

    __CONFIG _WDT_OFF&_PWRTE_ON&_XT_OSC&_LVP_OFF&_CP_OFF ; Configuración para el programador

    LIST p=16F877A
    INCLUDE <P16F877A.INC>

LCD_TEMP EQU 0x25   ; Definiciones necesarias
LCD_CHAR EQU 0x27   ; al utilizar la librería del LCD

val     EQU 0x20    ; Variable para el DELAY del ADC
val2    EQU 0x06    ; Variables para el DELAY del ENABLE LCD
val1    EQU 0x07

CBLOCK  0x32
Unidades            ; Variables para separar el resultado de ADC
Decenas             ; en valor BCD
Centenas
Resto
ENDC

  ORG 0x00

INICIO:
    clrf    PORTA        ;Limpia el puerto A
    clrf    PORTD       ;Limpia el puerto D
    bsf     STATUS,RP0
    bcf     STATUS,RP1    ;Cambio la banco 1
    clrf    TRISD        ;Configura PORTD como salida
    movlw   0x00
    movwf   ADCON1      ;Configura puerto A y E como analógicos
    movlw   3fh
    movwf   TRISA       ;Configura el puerto A como entrada
    movlw   0x00
    bcf     STATUS,RP0    ;Regresa al banco 0


;***************Zona de Programa***************

START
    call    LCD_CONFIG
    call    LCD_DATA
    CALL    RETARDO_150ms


      movlw b'11000001' ;Configuración ADCON0
      movwf ADCON0      ;ADCS1=1 ADCS0=1 CHS2=0 CHS1=0 CHS0= GO/DONE=0 - ADON=1

CICLO: bsf ADCON0,2      ;Conversión en progreso GO=1
       call DELAY1       ;Espera que termine la conversión
ESPERA btfsc ADCON0,2    ;Pregunta por DONE=0? (Terminó conversión)
       goto ESPERA       ;No, vuelve a preguntar
       movf ADRESH,0     ;Si

mostrar
        movlw   LCD_INICIO  ; Inicializa el LCD
        call    LCD_CMD     ; en la primera línea
        movlw   'T'         ; Envía letra a letra
        call    LCD_DATA    ; las palabras
```

```
72          movlw   'e'         ; Envía letra a letra
73          call    LCD_DATA    ; las palabras
74          movlw   'm'         ; Envía letra a letra
75          call    LCD_DATA    ; las palabras
76          movlw   'p'         ; Envía letra a letra
77          call    LCD_DATA    ; las palabras
78          movlw   '.'         ; Envía letra a letra
79          call    LCD_DATA    ; las palabras
80          movlw   '='         ; Envía letra a letra
81          call    LCD_DATA    ; las palabras
82
83        call leer   ;Llamada a rutina que obtine el
84                            ;valor de la temperatura a partir
85                            ;del  resultado del Conv a/D
86
87        movf Centenas,W   ;Imprime el dígito de las centenas
88        call LCD_DATA
89        movf Decenas,W    ;Imprime el dígito de las decenas
90        call LCD_DATA
91        movf Unidades,W   ;Imprime el dígito de las unidades
92        call LCD_DATA
93        movlw ' '
94        call LCD_DATA
95        movlw h'DF'       ;Imprime el simbolo "°"
96        call LCD_DATA
97        movlw 'C'
98        call LCD_DATA
99
100       call RETARDO_150ms
101       goto CICLO        ;Repite el ciclo de lectura ADC
102
103       ;Rutina que obtine el valor de la temperatura a partir del  resultado del Conv a/D
104
105  ;***************Zona de Subrutinas***************
106
107 LEER
108       clrf Centenas
109       clrf Decenas
110       clrf Unidades
111
112       movf ADRESH,W
113       addwf ADRESH,W     ;Dupilca el valor de ADRESH para
114       ;obtener un valor de temperatura real aprox
115       movwf Resto        ;Guarda el valor de ADRESH en Resto
116       ;Proceso de otención de valores BCD a traves de resta sucesivas.
117 CENTENAS1
118       movlw d'100'      ;W=d'100'
119       subwf Resto,W     ;Resto - d'100' (W)
120       btfss STATUS,C    ;Resto menor que d'100'?
121       goto DECENAS1     ;SI
122       movwf Resto       ;NO, Salva el resto
123       incf Centenas,1   ;Incrementa el contador de centenas BCD
124       goto CENTENAS1    ;Realiza otra resta
125 DECENAS1
126       movlw d'10'       ;W=d'10'
127       subwf Resto,W     ;Resto - d'10' (W)
128       btfss STATUS,C    ;Resto menor que d'10'?
129       goto UNIDADES1    ;Si
130       movwf Resto       ;No, Salva el resto
131       incf Decenas,1    ;Incrementa el contador de centenas BCD
132       goto DECENAS1     ;Realiza otra resta
133 UNIDADES1
134       movf Resto,W      ;El resto son la Unidades BCD
135       movwf Unidades
136       clrf Resto
137       ;Rutina que obtiene el equivalente en ASCII
138 OBTEN_ASCII
139       movlw h'30'
140       iorwf Unidades,f
141       iorwf Decenas,f
142       iorwf Centenas,f
143       return
144
145
146 ;Rutina que genera un Delay de 20 microSeg aprox.
```

```
147  ;para el Conv. A/D
148  DELAY1:
149         movlw h'30'
150         movwf val
151  Loop   decfsz val,1
152         goto Loop
153         return
154
155  INCLUDE  <LCD_int.inc>  ; Subrutinas de control del módulo LCD.
156       end
157
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 8 – Control iluminación PWM

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En esta práctica se trabajará con otro módulo interno del microcontrolador muy útil para las tareas de control, el módulo PWM. Este control es ampliamente conocido como "Pulse Weidth Modulation" y como su nombre indica consiste en variar el ciclo de trabajo de una señal pulsatoria de frecuencia constante.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa ensamblador MPLAB y el grabador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

Para practicar con el control PWM nos ayudaremos del led de alta luminosidad conectado al pin CCP1, aplicando así sobre éste la señal del PWM pudiendo variar su luminosidad de una forma más vistosa que con un led convencional. Además, y con tal de facilitar la observación de la variación, se adaptará el programa de la práctica anterior para medir el valor de la LDR y visualizarlo en el LCD.

También se puede conectar el jumper correspondiente al relé SSR que se encuentra conectado sobre el mismo pin que el led utilizado y conectar a éste una bombilla para realizar así el control PWM sobre una carga en alterna.

4. SOLUCIONARIO

```
1   ;*****************************************************************************************
    **
2   ;       Programa 8.1: Control luminosidad PWM
3   ;       Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
4   ;       Autor: Víctor Bueno Alvez
5   ;
6   ;       Este programa permite disminuir y aumentar el nivel de radiación de un led
    conectado al
7   ;       pin CCP1/RC3 mediante un control PWM contrlado por dos pulsadores conectadosa RA0
    y RA1.
8   ;       Microcontrolador: PIC16F877A
9   ;       Frecuencia: 4 MHZ
10  ;       Versión 1.0
11  ;*****************************************************************************************
    **
12
13  ;***************Zona de Definiciones***************
14      __CONFIG _WDT_OFF&_PWRTE_ON&_HS_OSC&_LVP_OFF&_CP_OFF
15
16      LIST p=16F877A
17      INCLUDE <P16F877A.INC>
18
19  cont    EQU 0x20
20  cont1   EQU 0x21
21  CTH     EQU 0x22
22  CTL     EQU 0x23
23
24      ORG 0X00
25
26  ;***************Zona de Programa***************
27
28  Inicio
29      bsf     STATUS,RP0
30      movlw   0x06        ; Configura el puerto A
31      movwf   ADCON1      ; como digital
32      movlw   0xFF
33      movwf   TRISA       ; Configura todos las patitas de PORTA como entradas
34      bcf     TRISC,5     ;Configura pines 5 ,6 y 7 como salida
35      bcf     TRISC,6
36      bcf     TRISC,7
37      movlw   0xFF        ; Configura parametros PWM
38      movwf   PR2
39      bcf     TRISC,2
40      bcf     STATUS,RP0
41      clrf    CCPR1L
42      bcf     CCP1CON,CCP1X
43      bcf     CCP1CON,CCP1Y
44      movlw   0x04
45      movwf   T2CON
46      bsf     CCP1CON,CCP1M3
47      bsf     CCP1CON,CCP1M2
48
49      clrf    CTL         ;Inicializa señal PWM
50      clrf    CTH
51
52  Esp0
53      btfsc   PORTA,0     ;Comprueba estado pulsador 1
54      call    incre       ;Si se ha pulsado incrementa PWM
55  Esp1                    ;Si no se ha pulsado continua
56      btfsc   PORTA,1     ;Comprueba estado pulsador 2
57      call    decre       ;Si se ha pulsado decreenta PWM
58      movf    CTL,W
59      movwf   CCPR1L
60
61      movlw   0xCF
62      andwf   CCP1CON,1
63      movlw   0x03
64      andwf   CTH,1
65      swapf   CTH,W
66      iorwf   CCP1CON,1
67
68      call    Retardo
69      goto Esp0
70
71  ;***************Zona de Subrutinas***************
```

1

```
72
73  incre                   ;Subrutina que aumenta el DC del PWM
74      incf    CTL,1
75      btfss   STATUS,Z
76      return
77      incf    CTH,1
78      return
79
80  decre                   ;Subrutina que disminuye el DC del PWM
81      decf    CTL,1
82      comf    CTL,W
83      btfss   STATUS,Z
84      return
85      decf    CTH,1
86      return
87
88  Retardo
89      clrf    cont1
90      clrf    cont
91  p1
92      decfsz  cont,1
93      goto    p1
94      decfsz  cont1,1
95      goto    p1
96      return
97
98      END
99
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 9 – Introducción a la programación en compilador
(Tutorial CCS)

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En esta primera práctica con el programa PIC C Compiler se pretende continuar trabajando de la misma forma que con ensamblador y por ello se comenzará también con una breve introducción de las diferentes herramientas del programa y se diseñará un programa de encendido de un led de una forma guiada.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa ensamblador MPLAB y el grabador PICkit2.

3. INTRODUCCIÓN PIC C Compiler

El compilador C de CCS ha sido desarrollado específicamente para PIC MCU, obteniendo la máxima optimización del compilador con estos dispositivos. Dispone de una amplia librería de funciones predefinidas, comandos de preprocesado y ejemplos. Además, suministra los controladores (drivers) para diversos dispositivos como LCD, convertidores AD, relojes en tiempo real, EEPROM serie, etc.

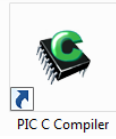
Un compilador convierte el lenguaje de alto nivel a instrucciones en código máquina y un cross-compiler, como en el caso de este compilador, es un compilador que funciona en un procesador (normalmente un PC) diferente al procesador objeto. Los programas son editados y compilados a instrucciones máquina en el entorno de trabajo del PC, el código máquina puede ser cargado del PC al sistema PIC desde cualquier programador pudiendo además, ser depurado desde el entorno de trabajo del PC.

El CCS es C estándar y, además de las directivas estándar (#include,etc.), suministra unas directivas específicas para PIC (#device,etc.) además de funciones específicas (bit_set(),etc.). Se suministra con un editor que permite controlar la sintaxis del programa.

4. DESARROLLO DE LA PRÁCTICA

Configuración PIC C Compiler

Iniciamos el programa utilizando el icono que aparece en el escritorio.



Una vez abierto el programa nos aparecerá una pantalla de color gris con las diferentes opciones que nos ofrece el programa. Para iniciar la escritura de nuestro programa deberemos utilizar el icono que muestra una carpeta abierta situado en la esquina superior izquierda.

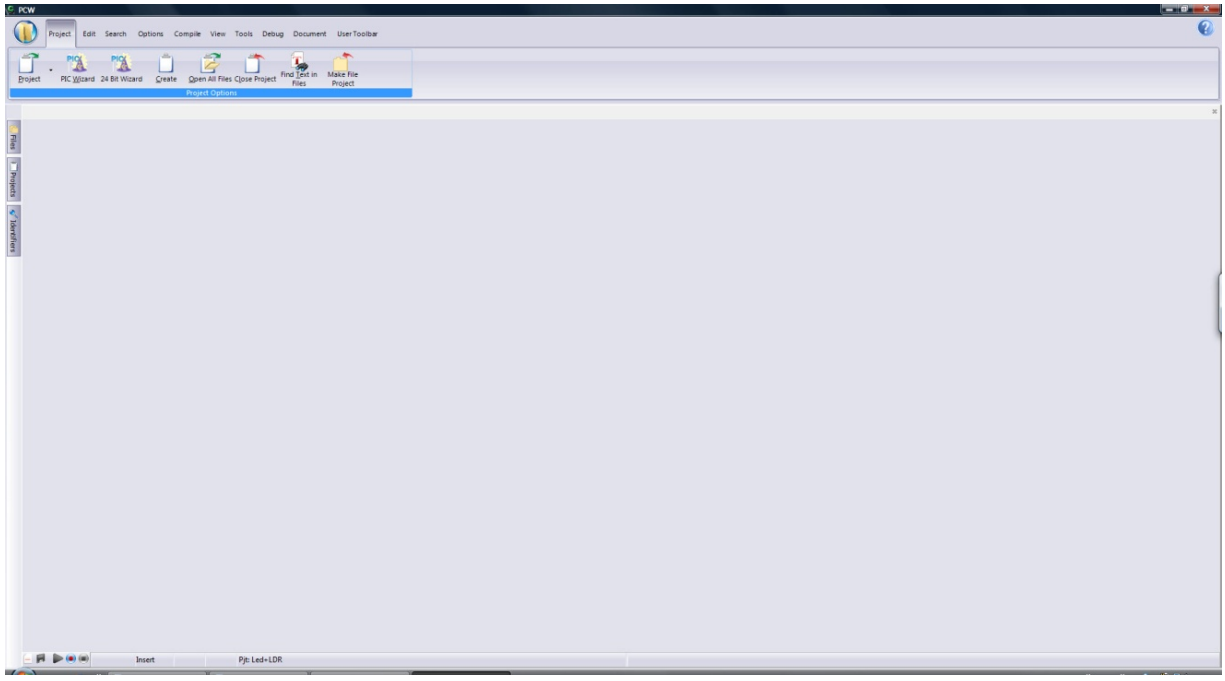


Figura *1. Pantalla Inicio*

Lo primero que haremos una vez abierto el compilador es ir al icono nombrado y crear un nuevo archivo fuente. Para realizar esta tarea el programa nos ofrece dos opciones, realizar el archivo de una forma manual seleccionando "Source File" con lo cual definiremos manualmente todos los registros de nuestro PIC o seleccinando la opción "Project Wizard" con una interfaz gráfica y algo más cómoda de utilizar. De todas formas, es recomendable trabajar manualmente con el programa sobretodo al inicarse con éste para tener control total acerca de la configuración del microcontrolador.

Por lo tanto, abrimos el menu archivos y seleccionamos la opción "New->Source File".

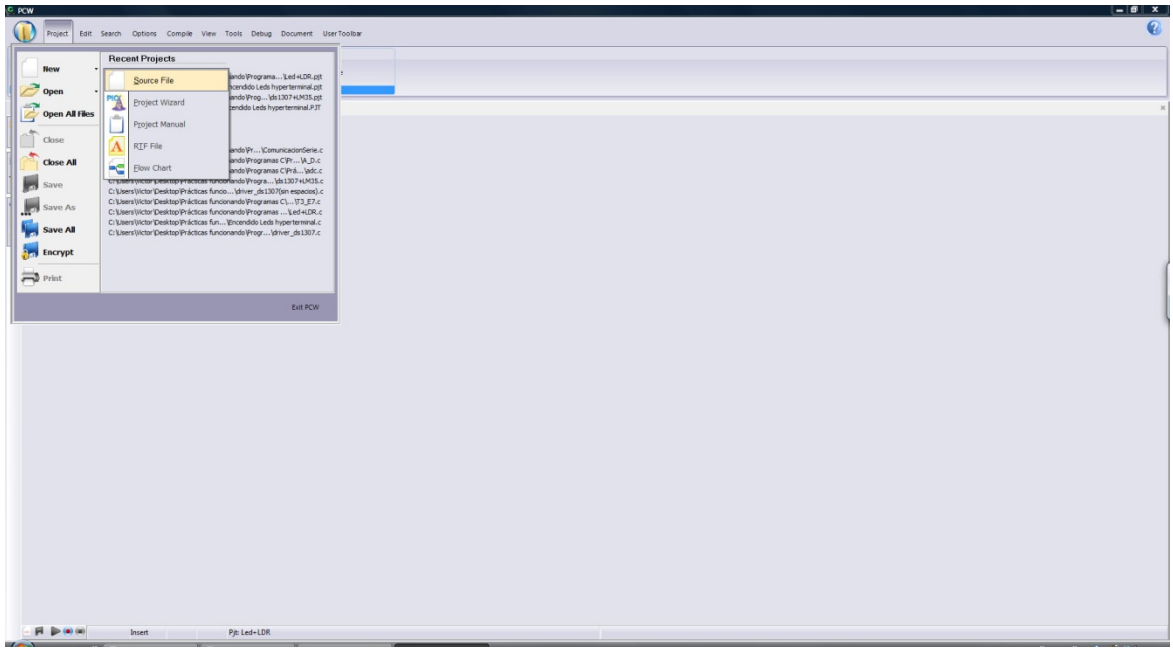


Figura 2. *Pestaña archivos*

Al crear un nuevo archivo fuente se no preguntará en que carpeta queremos guardarlo y como queremos nombrarlo, después de seleccionar la ubicación el programa nos creará el fichero y aparecerá una pantalla blanca en la cual escribiremos nuestro programa.

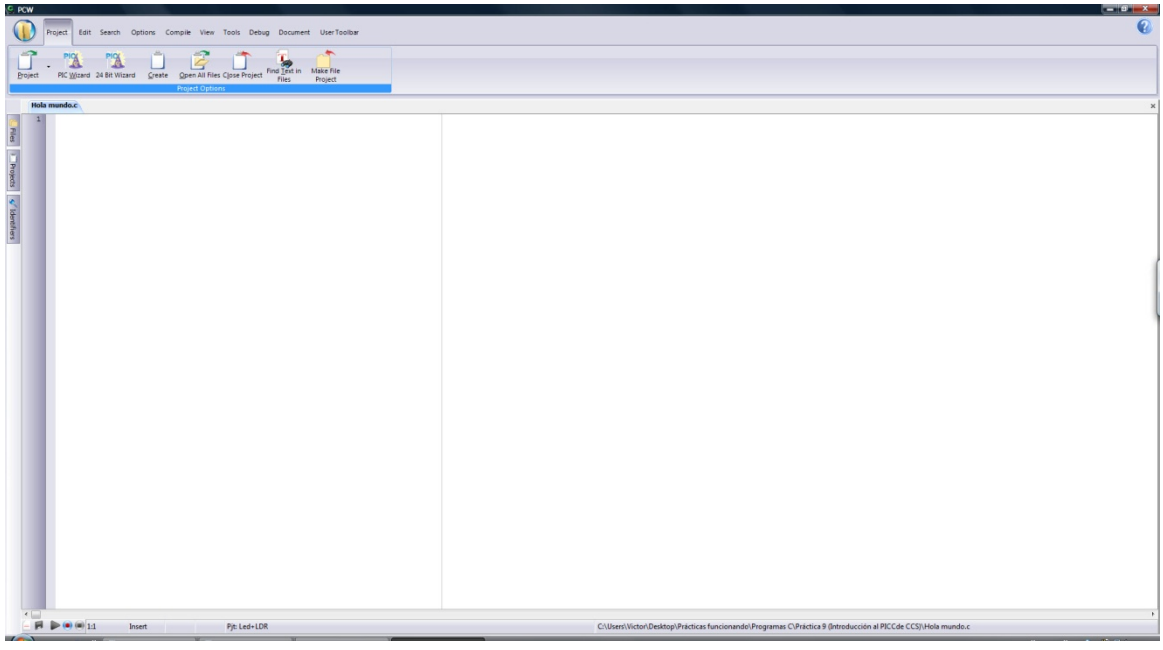


Figura 3. *Pantalla de trabajo*

Escritura del Programa

Una vez configurado el MPLAB procedemos a la escritura de nuestro propio programa. En nuestro caso utilizaremos un programa sencillo que se encarga de hacer parpadear un led conectado al puerto C del microcontrolador.

El programa utilizado es el siguiente:

```
1: //************************************************************************
2: //   Programa 9.1: Parpadeo 1 LED
3: //   Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
4: //   Autor: Víctor Bueno Alvez
5: //
6: //   Este programa hace parpadear un LED coenctado al pin C0.
7: //   Microcontrolador: PIC16F877A
8: //   Frecuencia: 4 MHZ
9: //************************************************************************
10:
11: //********************Zona de Definiciones*******************
12:
13: #include <16F877A.h>
14: #fuses XT, NOPROTECT, NOPUT, NOWDT, NOBROWNOUT, NOLVP, NOCPD
15: #use delay (clock=4000000)
16:
17: #BYTE TRISC=0x87
18: #BYTE PORTC=0x07
19:
20: //********************Zona de Programa*******************
21:
22: void main ()
23: {
24:    bit_clear(TRISC,0);
25:
26:    While(1){
27:       delay_ms(1000);
28:       bit_set (PORTC,0);
29:       delay_ms(1000);
30:       bit_clear (PORTC,0);
31:    }
32: }
33:
```

Figura 4. *Programa "Hola mundo"*

Lo primero que debemos hacer al redactar un programa es utilizar el fichero de cabecera donde se especifican las características del microcontrolador PIC:

#include <16f877a.h>

A continuación, se deben definir la velocidad del reloj del PIC y los puertos a utilizar. Además, en estas primeras líneas se puede escribir también la palabra de configuración para el programador.

#use delay (clock=4000000)

#byte puerto_c = 07

A medida que vamos escribiendo el programa descubrimos algunas de las propiedades que tiene el programa compilador y que son completamente configurables desde el panel de propiedades. Una propiedad muy atractiva para el usuario es el colorido de las diferentes palabras en función de su comportamiento, de esta forma nos podemos encontrar, por ejemplo, las directivas #include y #use de color rojo, las funciones while de color azul, etc... Otra de estas funciones se puede observar en la imagen anterior y es que cada vez que generamos un bucle como puede ser el while del programa anterior nos aparece un símbolo '-' a su izquierda que nos permite expandir o contraer este sector del programa, esta función es especialmente útil en programas de gran complejidad ya que permite centrar la atención en un único sector del programa.

Compilado del programa

Una vez finalizado el programa se debe proceder a su compilación para generar los archivos a copiar en nuestro microcontrolador. Para ello vamos a la barra de opciones y seleccionamos Compile dentro de la pestaña Compile. Los archivos que genera el programa pueden ser configurados para añadir algún otro formato como por ejemplo el formato *.cof para simulación de circuitos paso a paso con el programa de simulación Proteus.

Una vez seleccionada la opción de compilar nos aparecerá una ventana como la que se observa en la figura siguiente. En ella se muestra el proceso de compilación y la ocupación de memoria RAM y ROM en caso de que el programa esté correcto. En caso de que el programa tuviera algún error la compilación resultará fallida y nos aparecerá un listado de errores.

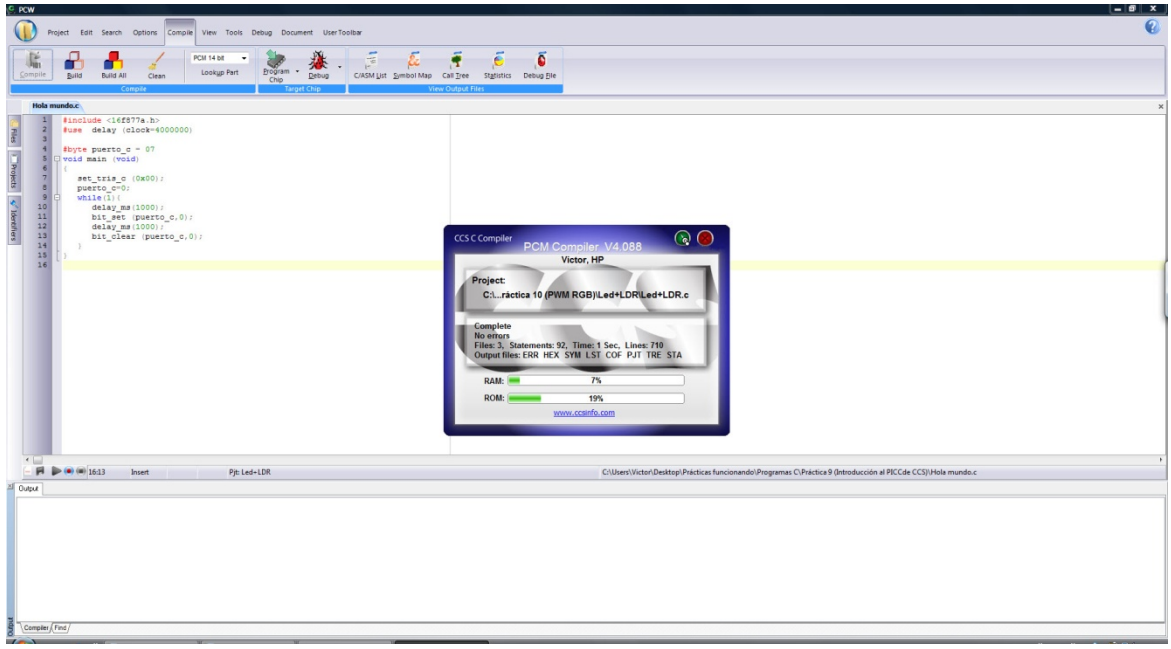


Figura 5. *Compilación*

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 10 – Control PWM

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En la siguiente práctica se pretende contrastar los dos lenguajes de programación utilizados en todo el conjunto de prácticas: ensamblador y C. De esta forma se implementará el último programa realizado en ensamblador pero en lenguaje C, para poder así ver las grandes diferencias entre los dos lenguajes de programación, la rapidez del lenguaje C frente a la optimización del ensamblador.

Esta práctica, por lo tanto, se dividirá en dos partes, una primera parte en donde se realizará un control PWM sobre un led, variando así su luminosidad, como el implementado en el programa realizado en ensamblador y una segunda parte en donde se deja libertad al alumno para probar las diferentes características del control PWM.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa compilador CCS y el depurador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

a) **Control iluminación PWM**

En esta primera parte de la práctica se pretende realizar el mismo programa implementado en la práctica 8 donde se controlaba la iluminación de un led de alta luminosidad mediante PWM. Para ello configuraremos el módulo de control PWM y la entrada analógica correspondiente a la LDR y vamos aumentando el Duty Cycle de la señal sucesivamente y realizando la medida correspondiente con la LDR mostrándola mediante el módulo LCD. Al trabajar con este lenguaje observamos que se acorta el código de programa, debido principalmente a la inclusión de los drivers de periféricos como el LCD y la facilidad de implementación de estructuras de programa. Por el contrario, si observamos la ocupación de memoria del microcontrolador en ambos programas, se puede comprobar la mayor optimización de recursos que se consigue con el ensamblador.

b) **Práctica libre**

Se deja libertad al lector para probar las diferentes funciones que nos ofrece el programa así como las funciones del módulo PWM del PIC. Además, se invita al lector a informarse acerca de la realización de varios PWM mediante los temporizadores internos del PIC, pudiendo así generar más señales PWM además de las dos incluidas directamente en el PIC.

4. SOLUCIONARIO

```
1: //************************************************************
2: //   Programa 10.1: Control PWM Led y medida LDR
3: //   Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
4: //   Autor: Víctor Bueno Alvez
5: //
6: //   Este programa hace aumentar el Duty Cicle de la señal CCP1 de 0 a 1 sucesivamente
7: //   variando así la luminosidad de un led y realizando a la vez medidas mediante
8: //   una LDR y visualizandolas en el LCD
9: //   Microcontrolador: PIC16F877A
10: //   Frecuencia: 4 MHZ
11: //************************************************************
12:
13: //*******************Zona de Definiciones*******************
14:
15: #include <16F877A.h>
16: #device adc=10
17: #fuses XT, NOPROTECT, NOPUT, NOWDT, NOBROWNOUT, NOLVP, NOCPD
18: #use delay (clock=4000000)
19: #include <lcd.c>
20: #BYTE TRISC=0x87
21: #BYTE PORTC=0x07
22: #BYTE TRISA=0x85
23: #BYTE PORTA=0x05
24: #BYTE TRISE=0x89
25: #BYTE PORTE=0x09
26:
27: //*******************Zona de Programa*******************
28:
29: void main( void )
30:
31: {
32:    int32 val;
33:    float temp;
34:    float lum;
35:    int   ciclo=0;
36:
37:    setup_Adc_ports (AN0);           //Canal analógico n°0
38:    setup_adc(ADC_CLOCK_INTERNAL);   //Selecciona fuente de reloj RC interno
39:
40:    setup_timer_2(T2_DIV_BY_1,255,1);
41:    setup_CCP1(CCP_PWM);
42:
43:    bit_clear(TRISC,2);
44:    lcd_init();
45:
46:   while ( TRUE )
47:
48:   {
49:       set_pwm1_duty(ciclo);
50:       bit_set(porte,2);
51:       set_adc_channel(1);          //Habilitamos canal 0
52:       delay_us(20);
53:       val=read_ADC();              //Leemos canal 0
54:       ;temp=(5.0*val)/(1024.0);    //Convertimos a temperatura (LM35) 10mV/°C
55:       lum=350-((5.0*val)/(10.240));//Convertimos a temperatura (LM35) 10mV/°C
56:
57:       set_adc_channel(0);          //Habilitamos canal 0
58:       delay_us(20);
59:       val=read_ADC();              //Leemos canal 0
60:       temp=(5.0*val)/(10.240);     //Convertimos a temperatura (LM35) 10mV/°
61:
62:       lcd_gotoxy(1,1);
63:
64:       printf(lcd_putc,"\Lum..: %01.1f lux",lum);
65:        printf(lcd_putc,"\nTemp.: %01.1f C",temp);
66:       delay_ms(300);
67:       ciclo++;
68:   }
69: }
70:
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 11 – Bus I2C: Reloj/Calendario DS1307

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En la siguiente práctica se inicia al estudiante en el bus de comunicaciones serie I2C. En esta primera práctica se utilizará el integrado DS1307, un reloj/calendario en tiempo real ampliamente conocido y que permitirá a nuestro microcontrolador saber exactamente en qué instante de tiempo está, y no únicamente la habilidad de medir intervalos de tiempo. Por lo tanto, el objetivo de esta práctica será el estudio de este bus de comunicaciones para poder desarrollar nuestra propia librería para el DS1307 o al menos comprender una de las que se encuentran en la red, y así poder controlar nuestro DS1307 para indicar la hora y el día exactos en un LCD.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa compilador CCS y el depurador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

Al igual que ocurría en ensamblador con el visualizador LCD, al leer la hoja de datos de este componente nos encontramos con que necesita unas rutinas de activación/desactivación, lectura/escritura, lo cual nos indica que deberemos proceder a realizar una librería de control. En este caso, se ha procedido a seleccionar una librería de una conocida web y a la adaptación de ésta para la aplicación en cuestión.

En cuanto al programa se debe tener en cuenta que el DS1307 dispone de una batería de soporte en caso de que le falte la alimentación, y que por lo tanto es capaz de guardar los datos correspondientes a la fecha y hora actual incluso con la tarjeta desconectada, por debemos considerar la inicialización de los registros correspondientes únicamente durante la primera programación, quitando estas líneas en modificaciones posteriores.

El objetivo de esta práctica es sencillo, comprender el funcionamiento del bus I2C así como de la librería correspondiente al DS1307 para posteriormente visualizar la fecha y la hora real en el módulo LCD. Para completar la práctica se pueden utilizar prácticas anteriores para, por ejemplo, mostrar la temperatura en la segunda línea o ampliar este programa para que encienda un led cada segundo o genere una acción a una hora determinada, por ejemplo, la activación de un relé.

La librería correspondiente al DS1307 utilizada en esta práctica se muestra en las siguientes páginas:

```
1: ////////////////////////////////////////////////////////////////////////////
2: ////                            libreria_ds1307.c                       ////
3: ////            Driver para reloj i2c de Dallas semiconductors          ////
4: ////            Adaptación por Víctor Bueno                             ////
5: ////                                                                    ////
6: ////////////////////////////////////////////////////////////////////////////
7:
8: ////////////////////////////////////////////////////////////////////////////////
9: //Prototipos del reloj:                                                      //
10: //    -ajustar_fecha_hora(): Escribimos via i2c la hora, fecha, mes, año...   //
11: //    -leer_hora_fecha(): Leemos via i2c todos los datos del reloj            //
12: //    -bin2bcd(): Rutina para comnvertir de binario a bcd                     //
13: //    -bcd2bin(): Rutina para convertir de bcd a binario                      //
14: ////////////////////////////////////////////////////////////////////////////////
15:
16: void ajustar_hora_fecha(void);
17: void leer_hora_fecha(void);
18:
19: void escribir_byte_ds1307(char direccion, char val);
20: char leer_byte_ds1307(char direccion);
21: char bin2bcd(char valor_binario);
22: char bcd2bin(char valor_bcd);
23:
24: ////////////////////////////////////////////////////////////////////////////////
25: //    Registros definiciones y variables usadas por el reloj, para almacenar  //
26: //    registros, comandos y funciones.                                        //
27: ////////////////////////////////////////////////////////////////////////////////
28:
29: /*Registros del reloj, los valores de éstos estan siempre en bcd
30: Según la posicion dentro del array almacenamos distintos registros:
31:
32:    SEGUNDOS             registros_ds1307[0]
33:    MINUTOS              registros_ds1307[1]
34:    HORAS                registros_ds1307[2]
35:    DIA DE LA SEMANA     registros_ds1307[3]
36:    FECHA                registros_ds1307[4]
37:    MES                  registros_ds1307[5]
38:    AÑO                  registros_ds1307[6]
39: */
40:
41: ////////////////////////////////////////////////////////////////////////////////
42: //     Registros del ds1307.                                                  //
43: ////////////////////////////////////////////////////////////////////////////////
44:
45: #define segundos 0
46: #define minutos 1
47: #define horas 2
48: #define dia_semana 3
49: #define fecha 4
50: #define mes 5
51: #define anio 6
52: #define registro_de_control 7
53:
54: ////////////////////////////////////////////////////////////////////////////////
55: /// Direcciones para leer/escribir al reloj.                                  //
56: ////////////////////////////////////////////////////////////////////////////////
57:
58: #define escribir_ds1307 0xd0
59: #define leer_ds1307 0xd1
60:
61: // Definimos un array global que contendrá los 7 registros del reloj...
62:
63: #define DS1307_DATE_TIME_BYTE_COUNT 7
64:
65: //para escribir en la memoria NVRAM qu etrae el reloj
66:
67: #define DS1307_NVRAM_START_ADDR 8
```

```
 68:
 69: // Con este valor desactivamos la señal de salida por el pin SQW
 70: // porque consume mucha bateria si está activada.
 71:
 72: #define DS1307_CONTROL_REG_INIT_VALUE 0x80
 73:
 74: //    Para activar el pin SQW a una frecuencia de 32768Hz
 75: //    Descomentaremos esta definicion
 76: //
 77: //#define DS1307_CONTROL_REG_INIT_VALUE 0x13
 78:
 79: ////////////////////////////////////////////////////////////////////////////////
 80: //    Con este array visualizamos el mes actual en formato texto.             //
 81: ////////////////////////////////////////////////////////////////////////////////
 82:
 83: const char meses1[12][4]=
 84:    {
 85:       "Ene",
 86:       "Feb",
 87:       "Mar",
 88:       "Abr",
 89:       "May",
 90:       "Jun",
 91:       "Jul",
 92:       "Ago",
 93:       "Sep",
 94:       "Oct",
 95:       "Nov",
 96:       "Dic"
 97:     };
 98:
 99: ////////////////////////////////////////////////////////////////////////////////
100: //    Esta es la variable global donde irán almacenados                        //
101: //    los registros del reloj.                                                 //
102: ////////////////////////////////////////////////////////////////////////////////
103:
104: char registros_ds1307[DS1307_DATE_TIME_BYTE_COUNT];
105:
106: ////////////////////////////////////////////////////////////////////////////////
107: //  Esta función convierte un valor comprendido entre 0 y 99 de binario a BCD //
108: ////////////////////////////////////////////////////////////////////////////////
109:
110: char bin2bcd(char valor_binario)
111: {
112:       char temp;
113:       char retval;
114:
115:       temp = valor_binario;
116:       retval = 0;
117:
118:       while(1)
119:          {
120:                // coge las decenas y les resta 10
121:                // para obtener unidades
122:                if(temp >= 10)
123:                   {
124:                      temp -= 10;
125:                      retval += 0x10;
126:                   }
127:                else
128:                   {
129:                      retval += temp;
130:                      break;
131:                   }
132:          }
133:    return(retval);
134: }
```

```
135:
136: ////////////////////////////////////////////////////////////////////////////
137: // Esta función convierte un valor comprendido entre 0 y 99 de BCD a binario  //
138: ////////////////////////////////////////////////////////////////////////////
139:
140: char bcd2bin(char valor_bcd)
141: {
142:       char temp;
143:       temp = valor_bcd;
144:       temp >>= 1;
145:       temp &= 0x78;
146:
147:    return(temp + (temp >> 2) + (valor_bcd & 0x0f));
148: }
149:
150: ////////////////////////////////////////////////////////////////////////////
151: //    Con esta función escribimos la hora y la fecha mediante i2c en el reloj //
152: //    DS1307.                                                                 //
153: //                                                                            //
154: //    Primero convertiremos los valores de binario a BCD para poder           //
155: //    almacenarlos, seguidamente desactivaremos las interrupciones            //
156: //    para no interrumpir la transmision, y los enviaremos con las            //
157: //    funciones del compilador para el manejo de i2c.                         /7
158: ////////////////////////////////////////////////////////////////////////////
159:
160: void ajustar_hora_fecha(void)
161: {
162:       char i;
163:       for(i = 0; i < 7; i++)
164:          {
165:             //registros_ds1307 = bin2bcd(registros_ds1307);**************************
166:            registros_ds1307[i] = bin2bcd(registros_ds1307[i]);
167:          }
168:
169:       // Existen 2 bits de control en los registros del reloj, hay uno
170:       //para parar el reloj en el bit 7 del registro de los segundos.
171:       // tenemos que asgurarnos que éste es cero para que el reloj funcione
172:       //hay otro bit para seleccionar modo 12/24 horas, para poner en modo de
173:       // 24 horas, lo ponemos a cero,sino a 1
174:
175:       registros_ds1307[segundos] &= 0x7f;
176:       registros_ds1307[horas] &= 0x3f;
177:
178:       // escribimos los datos en bcd al array global
179:
180:       disable_interrupts(GLOBAL);
181:
182:       i2c_start();
183:       i2c_write(escribir_ds1307);
184:
185:       // se empieza leyendo los segundos
186:
187:       i2c_write(segundos);
188:
189:       // y luego se escriben 7 bytes más....
190:
191:       for(i = 0; i < 7; i++)
192:          {
193:             i2c_write(registros_ds1307[i]);
194:          }
195:
196:       // despues de ajustar los registros
197:       // modificamos el registro de control
198:
199:       i2c_write(DS1307_CONTROL_REG_INIT_VALUE);
200:       i2c_stop();
201:       enable_interrupts(GLOBAL);
```

```
202: }
203:
204: ////////////////////////////////////////////////////////////////////////////
205:
206: //    Con esta función leemos la hora y la fecha mediante i2c del array       //
207:
208: //    global que contiene todos los registros en BCD                          //
209:
210: //                                                                            //
211:
212: //    Primero desactivaremos las interrupciones para evitar que interrumpan   //
213:
214: //    la transmision, seguidamente usamos las funciones del compialdor        //
215:
216: //    para el manejo de i2c y por último convertiremos de BCD a binario       //
217:
218: //    para poder visualizarlas correctamente.                                 //
219:
220: ////////////////////////////////////////////////////////////////////////////
221:
222: //    Los valores almacenados en el array podrán estar comprendidos entre:    //
223:
224: //                                                                            //
225:
226: //    Segundos       0-59                                                     //
227:
228: //    Minutos        0-59                                                     //
229:
230: //    Horas          0-23                                                     //
231:
232: //    Dia semana     1-7                                                      //
233:
234: //    Dia            1-31                                                     //
235:
236: //    Mes            1-12                                                     //
237:
238: //    Año            00-99 (a partir del año 2000)                            //
239:
240: ////////////////////////////////////////////////////////////////////////////
241:
242: void leer_hora_fecha(void)
243: {
244:       char i;
245:
246:       disable_interrupts(GLOBAL);
247:
248:       i2c_start();
249:       i2c_write(escribir_ds1307);
250:
251:       // empezamos leyendo los segundos
252:
253:       i2c_write(segundos);
254:
255:       i2c_start();
256:       i2c_write(leer_ds1307);
257:
258:       // leemos los 7 bytes restantes. hacemos mascaras para los
259:       //bits que no usamos
260:
261:       registros_ds1307[segundos] = i2c_read() & 0x7f;
262:       registros_ds1307[minutos] = i2c_read() & 0x7f;
263:       registros_ds1307[horas] = i2c_read() & 0x3f;
264:       registros_ds1307[dia_semana] = i2c_read() & 0x07;
265:       registros_ds1307[fecha] = i2c_read() & 0x3f;
266:       registros_ds1307[mes] = i2c_read() & 0x1f;
267:       registros_ds1307[anio] = i2c_read(0);
268:
```

```
269:       i2c_stop();
270:
271:       enable_interrupts(GLOBAL);
272:
273:       // convertimos de bcd a binario
274:
275:       for(i = 0; i < 7; i++)
276:          {
277:             //registros_ds1307 = bcd2bin(registros_ds1307);**************************
278:            registros_ds1307[i] = bcd2bin(registros_ds1307[i]);
279:          }
280: }
281:
282: ////////////////////////////////////////////////////////////////////////////////
283:
284: //    Funcion para leer en un registro de la memoria NVRAM que trae           //
285:
286: //    incorporada el reloj DS1307.                                            //
287:
288: ////////////////////////////////////////////////////////////////////////////////
289:
290: char leer_byte_ds1307(char direccion)
291: {
292:       char retval;
293:
294:       disable_interrupts(GLOBAL);
295:
296:       i2c_start();
297:       i2c_write(escribir_ds1307);
298:       i2c_write(direccion);
299:
300:       i2c_start();
301:       i2c_write(leer_ds1307);
302:       retval = i2c_read(0); // no esperamos por ack
303:       i2c_stop();
304:
305:       enable_interrupts(GLOBAL);
306:
307:    return(retval);
308: }
309:
310: ////////////////////////////////////////////////////////////////////////////////
311:
312: //    Funcion para escribir en un registro de la memoria NVRAM que trae       //
313:
314: //    incorporada el reloj DS1307.                                            //
315:
316: ////////////////////////////////////////////////////////////////////////////////
317:
318: void escribir_byte_ds1307(char direccion, char val)
319:
320: {
321:       disable_interrupts(GLOBAL);
322:
323:       i2c_start();
324:       i2c_write(escribir_ds1307);
325:       i2c_write(direccion);
326:       i2c_write(val);
327:       i2c_stop();
328:
329:       enable_interrupts(GLOBAL);
330: }
331:
```

4. SOLUCIONARIO

```
1: //*************************************************************
2: //   Programa 11.1: Control RTC DS1307
3: //   Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
4: //   Autor: Víctor Bueno Alvez
5: //
6: //   Este programa nos permite visualizar la hora, la fecha y la temperatura
7: //   gracias al control del RTC mediante la librería <driver_ds1307.c> y la
8: //   medida de tenperatura mediante el LM35conectado al canal AN0
9: //   Microcontrolador: PIC16F877A
10: //   Frecuencia: 4 MHZ
11: //*************************************************************
12:
13: //*****Zona de Definiciones
14: #include <16F877A.h>
15: #device adc=10
16: #fuses XT, NOPROTECT, NOPUT, NOWDT, NOBROWNOUT, NOLVP, NOCPD
17: #use delay (clock=4000000)
18: #use i2c(Master, SDA=PIN_C4, SCL=PIN_C3)
19: #include <lcd.c>
20: #include <driver_ds1307.c>
21:
22: //*****Zona de Programa
23: void main( void )
24: {
25:    int32 val;
26:    float temp;
27:    setup_Adc_ports (AN0);           //Canal analógico nº0
28:    setup_adc(ADC_CLOCK_INTERNAL);   //Selecciona fuente de reloj RC interno
29:    lcd_init();
30:
31: //escribimos la fecha y la hora actual en los registros del DS1307
32:
33: //      registros_ds1307[0] = 00; //  segundos
34: //      registros_ds1307[1] = 54; // minutos
35: //      registros_ds1307[2] =10;  // horas
36: //      registros_ds1307[3]= 0;
37: //      registros_ds1307[4]= 21;  //Dia
38: //      registros_ds1307[5] = 12; // Mes
39: //      registros_ds1307[6] = 10; // año
40:
41: // Escribir valores al DS1307
42:
43: //      ajustar_hora_fecha();
44:
45: //      lcd_putc("\f");
46:
47:   while ( TRUE )
48:   {
49:       leer_hora_fecha();
50:       set_adc_channel(0);     //Habilitamos canal 0
51:       delay_us(20);
52:       val=read_ADC();         //Leemos canal 0
53:       temp=(5.0*val)/(10.24);  //Convertimos a temperatura (LM35) 10mV/ºC
54:       lcd_gotoxy(1,1);
55:       printf(lcd_putc,"%02u:%02u:%02u/%02u%s%02u\n",registros_ds1307[horas],registros_ds
56:       printf(lcd_putc,"Temp: %01.1f C",temp);
57:       delay_ms(500);
58:   }
59: }
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 12 – Bus I2C: ADC/DAC PCF8591

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En la siguiente práctica se continuará utilizando el bus de comunicaciones I2C, en este caso con un convertidor AD/DA. Para ello se realizarán dos programas distintos, para observar así los dos modos de funcionamiento. En primer lugar se configurará el PCF8591 en modo DA y generaremos una rampa en su pin AOUT y para completar la práctica se realizará un programa que a partir de la rampa generada por el integrado mida dicha señal y la visualice en el LCD mediante uno de sus cuatro canales de entrada.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa compilador CCS y el depurador PICkit2.

3. DESARROLLO DE LA PRÁCTICA

En esta práctica se pretende realizar un control sencillo mediante el integrado PCF8591, utilizando su modo AD y DA. Para ello, se ha generado una señal rampa mediante este integrado, se podría haber escogido cualquier otra señal pero se ha escogido esta debido a su facilidad de generación, también se podría implementar una señal cuadrada sin aumentar demasiado el nivel de dificultad pero esta señal se considera de más fácil entendimiento debido a su linealidad, lo que además, facilita su visualización, así como la detección de posibles problemas. Por todo lo comentado, se recomienda al lector iniciarse al uso de este integrado con este tipo de señal pudiendo probar otros distintos una vez comprendido su funcionamiento.

a) **PCF8591 modo DA**

En esta primera parte de la práctica configuraremos el integrado PCF8591 como convertidor DA e iremos enviándole valores sucesivos para generar así una rampa.

b) **PCF8591 modo AD**

En esta segunda parte partiremos del programa realizado en el apartado anterior para generar una señal rampa, medir su valor de tensión y visualizarlo en el LCD. Para ello iremos conmutando entre los dos modos de trabajo del integrado mediante las diferentes palabras de control.

4. SOLUCIONARIO

```
1: //*************************************************************
2: //   Programa 12.1: Control DAC I2C PCF8591
3: //   Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
4: //   Autor: Víctor Bueno Alvez
5: //
6: //   Este programa nos permite generar una rampa desde 0 a 5V a través del
7: //   pin AOUT de PCF8591, en este caso no es necesario crear una libreria
8: //   siendo necesario enviar únicamente el valor de la señal deseada
9: //   Microcontrolador: PIC16F877A
10: //   Frecuencia: 4 MHZ
11: //*************************************************************
12:
13: //*****Zona de Definiciones
14: #include <16f877a.h>
15: #fuses XT,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
16: #use delay(clock=4000000)
17: #use standard_io(b)
18: #use i2c(master,sda=PIN_C4, scl=PIN_C3) // Configuración del I2C como Master y los pines
19: int  analogico=0x00;
20:
21: //*****Zona de Programa
22: void main() {
23:
24:    i2c_start();               // Inicio de la comunicación I2C
25:    i2c_write(0b10011110);     // Envío Dirección I2C del PCF8591
26:    i2c_write(0b01000000);     // Envío Configuración del PCF8591 para Conv. DA
27:    do {
28:       i2c_write(++analogico); // Envío Valor digital 0x00->0V, 0xFF->Vcc
29:       delay_ms(300);
30:    } while (TRUE);
31: }
32:
33:
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 13 – Control supervisado hyperterminal

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

Con esta práctica se pretende trabajar con el puerto de comunicación serie del microcontrolador. Con este objetivo se conectará la tarjeta entrenadora con el puerto COM del ordenador mediante un cable serie y programaremos el microcontrolador para que nos envíe los datos adquiridos por el convertidor A/D utilizando el Hypeterminal, un programa visualizador del estado del puerto serie para Windows, el cual nos mostrará los datos en la pantalla del ordenador.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa compilador CCS y el depurador PICkit2.
- Cable comunicación serie.

3. DESARROLLO DE LA PRÁCTICA

a) Introducción

Antes de comenzar con la programación del microcontrolador es conveniente hacer un repaso sobre dos aspectos importantes en la realización de esta práctica. De una banda debemos entender el funcionamiento del integrado max232, encargado de comunicar el PIC con el ordenador y por otro lado comprender el programa Hyperterminal para Windows además de la teoría referente a la comunicación serie, la cual se considera entendida una vez iniciada esta práctica..

Lo primero que debemos tener en cuenta es que es imposible comunicar un microcontrolador directamente con un PC, aunque cuente con dos pines destinados para tal fin, esto es así puesto que el microcontrolador trabaja con niveles de señal TTL y el puerto RS232 trabaja con tensiones mayores, lo cual nos indica que deberemos utilizar algún circuito complementario para tal fin. El circuito utilizado para comunicar el microcontrolador con el PC a través del puerto serié es el integrado max232 el cual incluye internamente 2 conversores de nivel de TTL a rs232 y otros 2 de rs232 a TTL con lo que en total podremos manejar 4 señales del puerto serie del PC, las cuáles acostumbran a ser TX y RX para la comunicación además de RTS y CTS para el control de flujo. Éstas dos últimas son las usadas para el protocolo handshaking pero no es imprescindible su uso.

Por otro lado, una vez conectado el microcontrolador al PC con la señales adaptadas correctamente, necesitamos algún programa instalado en el PC que escuche el puerto serie al cual se encuentra conectado el microcontrolador y actúe en consecuencia, en este caso, por ejemplo, mostrando los datos por pantalla, este es el cliente Hyperterminal de Windows. Este programa, que se incluía en las primeras versiones de Windows y que deberemos descargar gratuitamente a partir de la versión de Windows Vista, permite realizar configurar una comunicación por puerto serie definiendo su velocidad, bits de parada, paridad, etc. Por lo tanto es un programa muy útil para iniciarse en la comunicación serie con PIC. Como se verá a continuación su configuración es realmente sencilla.

b) Programa para el microcontrolador

En esta práctica se pretende diseñar un programa sencillo para el microcontrolador que se encargue de enviar un dato adquirido por el microcontrolador a través del puerto serie. De esta forma podemos enviar el valor de tensión en los potenciómetros, temperatura medida por el LM35 o cualquier valor que se nos ocurra de nuestra tarjeta entrenadora. Estos valores se pueden enviar de una forma continuada intercalando un signo de separación, o utilizando una nueva línea de pantalla para cada número.

c) Configuración Hyperteminal Windows

Una vez programado el microcontrolador debemos crear un programa en Windows que permita la comunicación. En este caso, como se ha comentado anteriormente se utilizará un programa ya integrado en Windows, pero se podría realizar uno nuevo, basándonos en Visual Basic, Labview u otros lenguajes de programación como veremos en las siguientes prácticas. Lo que siempre deberemos tener en cuenta es configurar ambos dispositivos con las mismos parámetros para que puedan entenderse: velocidad, nº bits, paridad.

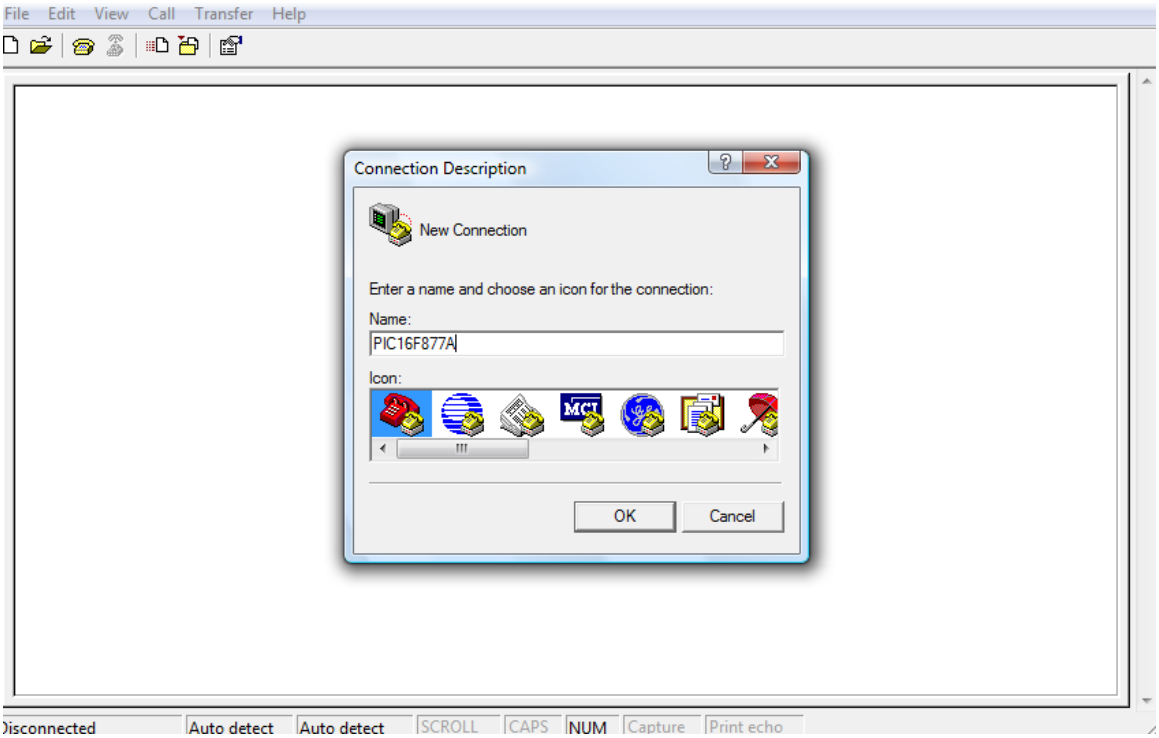


Figura 1. *Abrimos una nueva conexión en el hyperterminal*

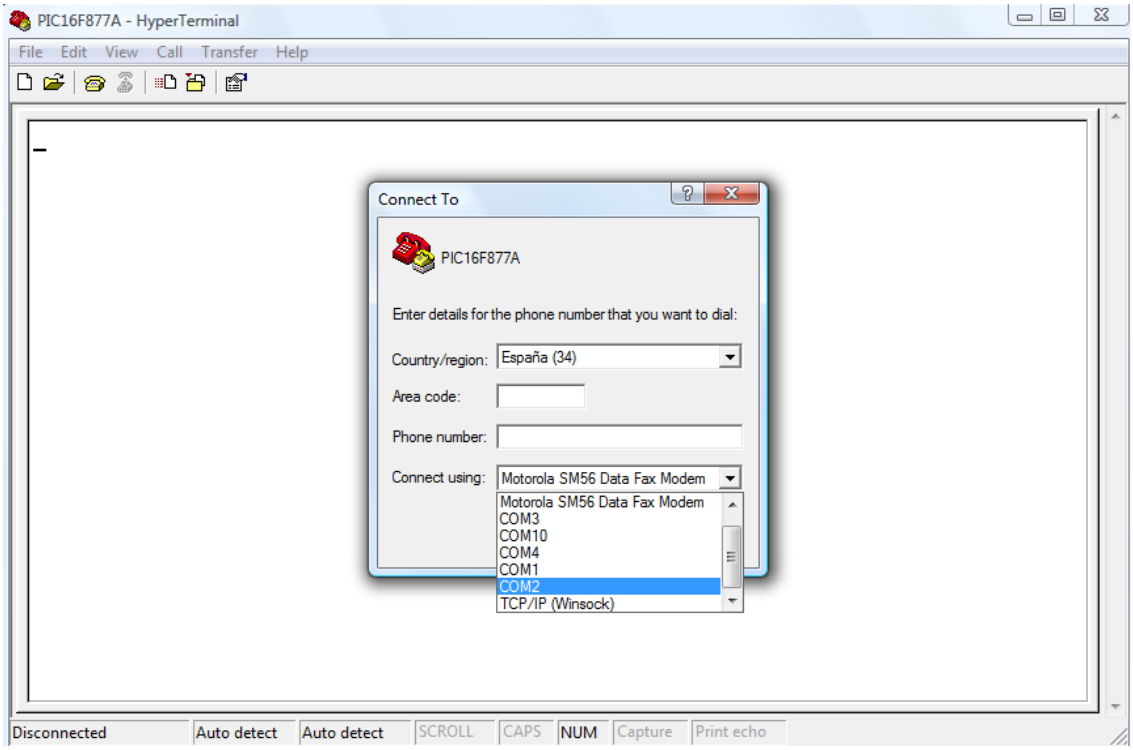


Figura 2. *Elegimos el puerto de comunicaciones*

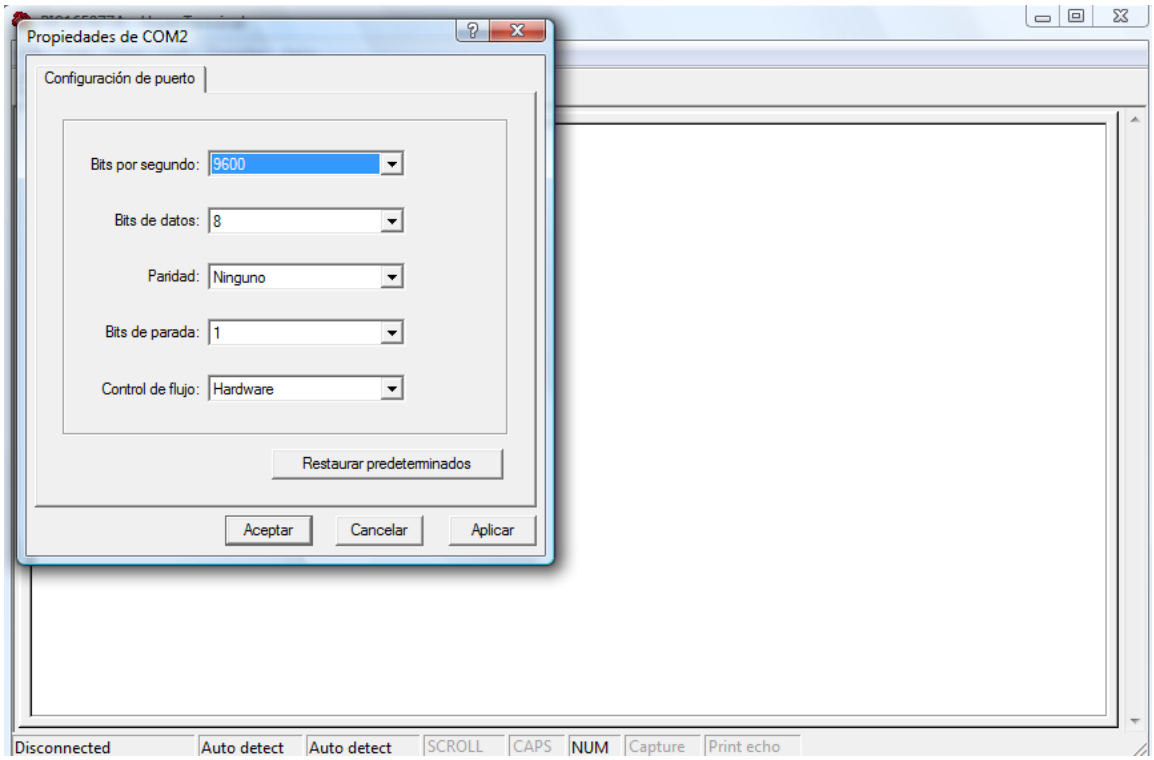


Figura 3. *Configuramos el puerto de la misma forma que el microcontrolador*

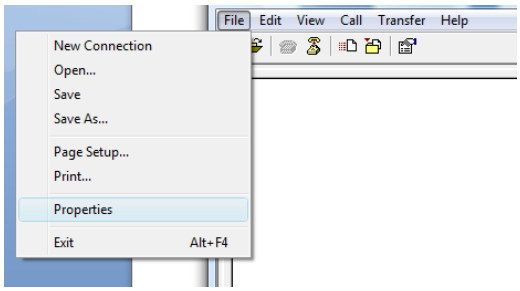


Figura 4. *Agregamos avance de línea (1)*

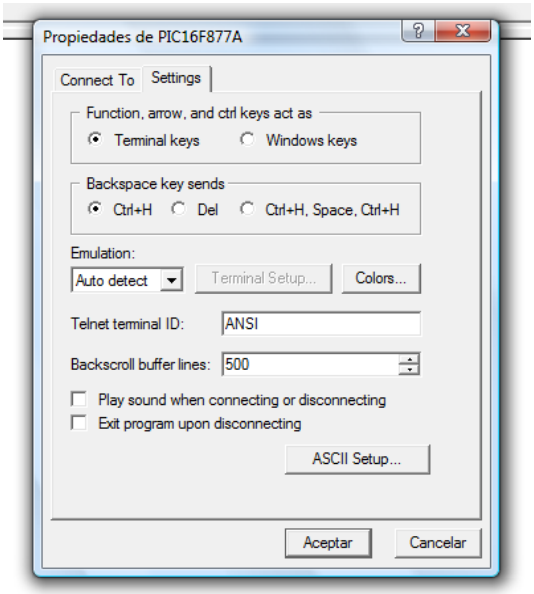


Figura 5. *Agregamos avance de línea (2)*

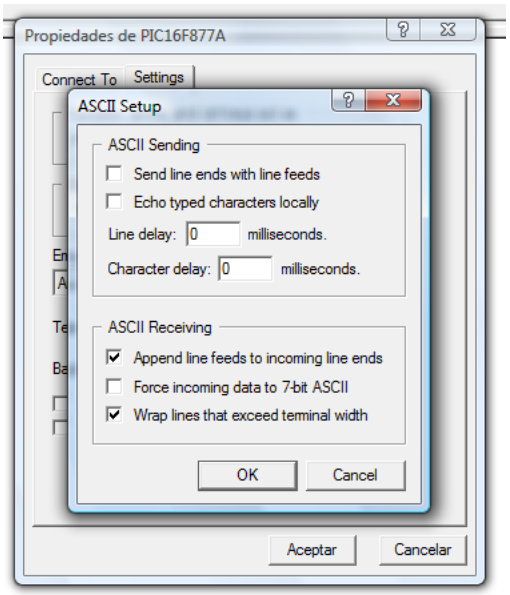


Figura 6. *Agregamos avance de línea (3)*

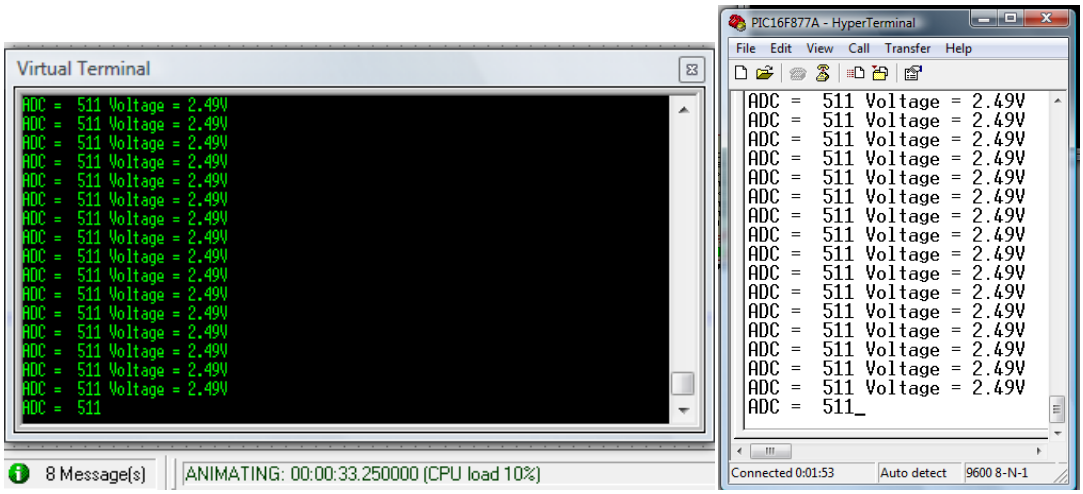


Figura 7. *Circuito funcionando en proteus*

4. SOLUCIONARIO

```
1: //////////////////////////////////////////////////////////////////////////
2: //   Programa 13.1: Comunicacion serie Hyperterminal
3: //   Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
4: //   Autor: Víctor Bueno Álvez
5: //
6: //   Este programa nos permite visualizar la temperatura medida por el PIC
7: //   mediante el hyperterminal de windows comunicandose por el puerto RS232.
8: //   Microcontrolador: PIC16F877A
9: //   Frecuencia: 4 MHZ
10: //   Versión: 1.0
11: //////////////////////////////////////////////////////////////////////////
12:
13: //************************Zona de Definiciones*****************************
14:
15: #include <16F877a.h>
16: #device adc=10
17: #fuses XT,NOWDT
18: #use delay(clock=4000000)
19: #use rs232(baud=9600, xmit=pin_c7, rcv=pin_c6, bits=8, parity=N)
20: #include <LCD.C>
21:
22:
23: //************************Zona de Programa*********************************
24: void main() {
25:    int16 v;
26:    float temp;
27:    setup_adc_ports(AN0);
28:    setup_adc(ADC_CLOCK_INTERNAL);
29:    lcd_init();
30:    while (1) {
31:       set_adc_channel(0);
32:       delay_us(10);
33:       v = read_adc();
34:       temp = 5.0 * v / 10.24;
35:       printf(lcd_putc, "\fADC : %4ld", v);
36:       printf(lcd_putc, "\nTemperatura : %01.2fV", temp);
37:       printf("ADC : %4ld ", v);
38:       printf("Temperatura : = %01.2fV\r", temp);
39:       delay_ms(150);
40:       }
41: }
```

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 14 – Control supervisado LabVIEW

Víctor Bueno Álvez

Enero 2011

1. OBJETIVOS

En la siguiente práctica se pretende utilizar todos los conocimientos obtenidos en las prácticas anteriores para crear un sistema de control supervisado. El tema es libre, pero debería consistir, como mínimo, en visualizar una señal medida por el microcontrolador, de la misma forma que la práctica anterior pero realizando el programa en labVIEW, además ésta deberá ser graficada.

Con esta práctica se pretende continuar trabajando con el puerto de comunicación serie del microcontrolador, substituyendo en este caso el programa Hypeterminal por uno creado en LabVIEW. Además utilizaremos una comunicación con control de flujo por hardware, es decir, con 4 señales de comunicación y no con dos como hemos hecho anteriormente. De esta forma está permitido enviar cualquier tipo de valor por el puerto ya que no es necesario reservar ninguno para el control de flujo por software, ya que esto se realizará mediante las líneas RTS y CTS. Esta comunicación está en desuso pero es útil conocerla ya que en aplicaciones concretas es necesaria o indispensable su utilización.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa compilador CCS y el depurador PICkit2.
- Software de desarrollo LabVIEW.
- Cable comunicación RS232.

3. DESARROLLO DE LA PRÁCTICA

a) Introducción

LabVIEW es una herramienta gráfica para pruebas, control y diseño mediante la programación. El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje Gráfico. Su principal característica es la facilidad de uso, lo que lo hace válido para programadores profesionales y personas con pocos conocimientos en programación. Los programas en LabVIEW son llamados instrumentos virtuales (VIs)

El control de flujo se puede hacer o por las líneas de control, método que se suele utilizar en un interfaz de comunicación de datos, o reservando caracteres de control al comienzo y al final del flujo de la señal (códigos para XON/XOFF). Las líneas destinadas al control de flujo en RS 232 son RTS (petición de envío) y CTS (listo para enviar). Éste método es denominado como "control de flujo por hardware". El otro método donde se utilizan códigos de control (XON/XOFF) se denomina generalmente como "control de flujo por software".

b) Programa para el microcontrolador

A continuación se muestra el código del programa del microcontrolador. En la cabecera del programa se inicia la comunicación RS232, el módulo LCD y se activa el canal AN0, para posteriormente entrar en un bucle infinito en donde se envía la petición para el envío (PORTC,0) y posteriormente se procede a enviar el dato cuando el PC acepta el envío (PORTC,1).

```
1: //*****************************************************************
2: //   Programa 13.1: Comunicación RS232
3: //   Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
4: //   Autor: Víctor Bueno Alvez
5: //
6: //   Este programa nos permite visualizar la temperatura y el valor del ADC
7: //   en la pantalla del programa Hypeterminal de windows
8: //   Microcontrolador: PIC16F877A
9: //   Frecuencia: 4 MHZ
10: //*****************************************************************
11:
12: //*****Zona de Definiciones
13:
14: #include <16F877A.h>
15: #device adc=10
16: #FUSES XT,NOWDT
17: #use delay(clock=4000000)
18: #use rs232(baud=9600, xmit=pin_c6, rcv=pin_c7, bits=8, parity=N)
19: #include <LCD.C>
20: #BYTE TRISC =0X87
21: #BYTE PORTC =0X07
22:
23: float ch;
24: float p,q;
25:
26: void main() {
27:
28:    int flag=0;
29:
30:    bit_set(TRISC,1);
31:    bit_clear(TRISC,0);
32:    bit_clear(TRISC,2);
33:
34:    setup_adc_ports(AN0);
35:    setup_adc(ADC_CLOCK_INTERNAL);
36:    lcd_init();
37:
38:    for (;;) {
39:       set_adc_channel(0);
40:       delay_us(10);
41:       q = read_adc();
42:
43:       bit_set(PORTC,0);
44:
45:       if((bit_test(PORTC,1)==1)){
46:
47:          bit_clear(PORTC,0);
48:          delay_ms(20);
49:          printf("%01.2f",q);  // El \r permite cambiar de línea.
50:          delay_ms(20);
51:          printf(lcd_putc, "\nADC = %01.2fV", q);
52:          while((bit_test(PORTC,1)==1)){
53:             bit_set(PORTC,0);
54:
55:          }
56:       }
57:
58:       }
59: }
```

c) Programa Labview

Lo primero que nos encontramos al ejecutar el programa LabVIEW 2009 es un menú de inicio como el que se muestra en la figura siguiente. En este menú podemos encontrar las diferentes opciones que nos ofrece el programa para diseñar un nuevo proyecto, ya sea seleccionando un VI, proyecto nuevo, o partiendo de un ejemplo de los muchos que contiene LAbVIEW.

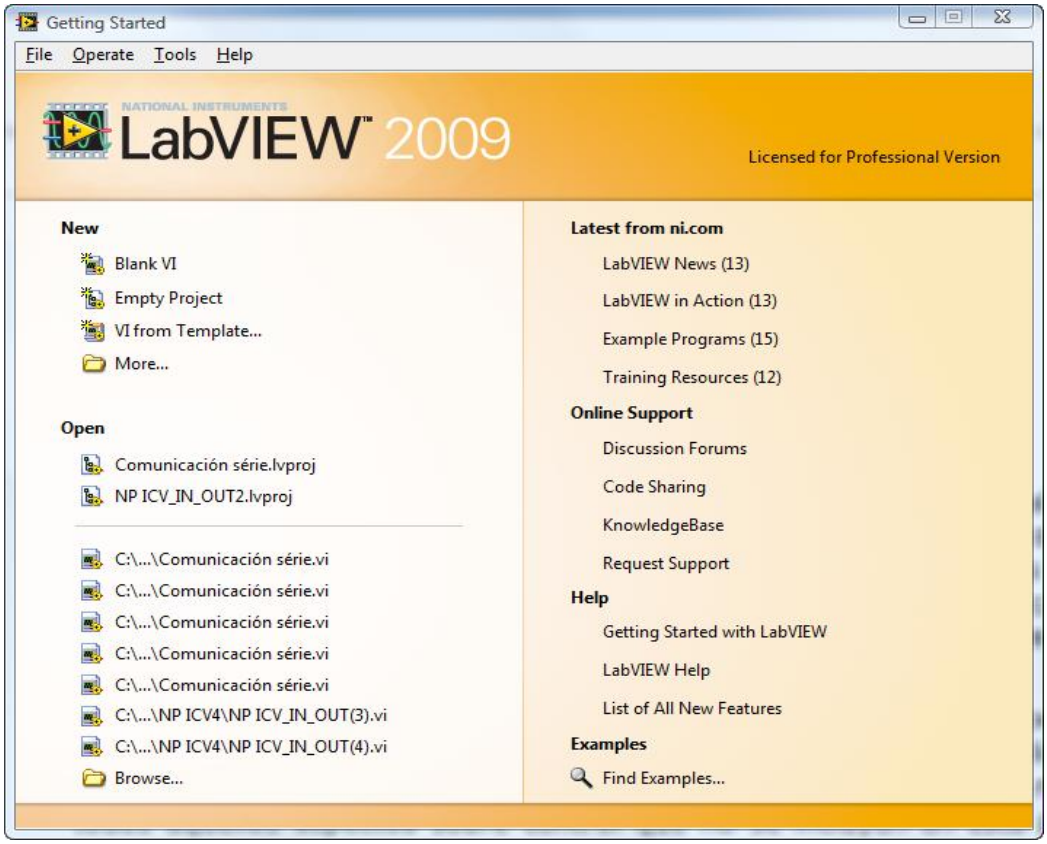


Figura 1. *Pantalla Inicio*

En este caso seleccionaremos la opción "Blank VI" para proceder a la implementación de nuestro nuevo instrumento virtual.

Al ejecutar esta opción nos aparecerán dos nuevas ventanas como las que se observan en las siguientes figuras, la primera corresponde al panel frontal de nuestro instrumento y es sobre la que podemos trabajar de una forma visual colocando indicadores, interruptores y todo aquello que deseemos. La segunda pantalla corresponde al diagrama de bloques y es aquí donde se encuentra la mayor potencia de este software, ya que nos permite realizar instrumentos complejos utilizando para ello bloques y conexiones entre ellos mediante hilos.

Por lo tanto, la programación en LabVIEW se fundamenta en la utilización de estas dos pantallas, la primera nos muestra el aspecto del instrumento y la segunda nos ofrece la potencia de programación.

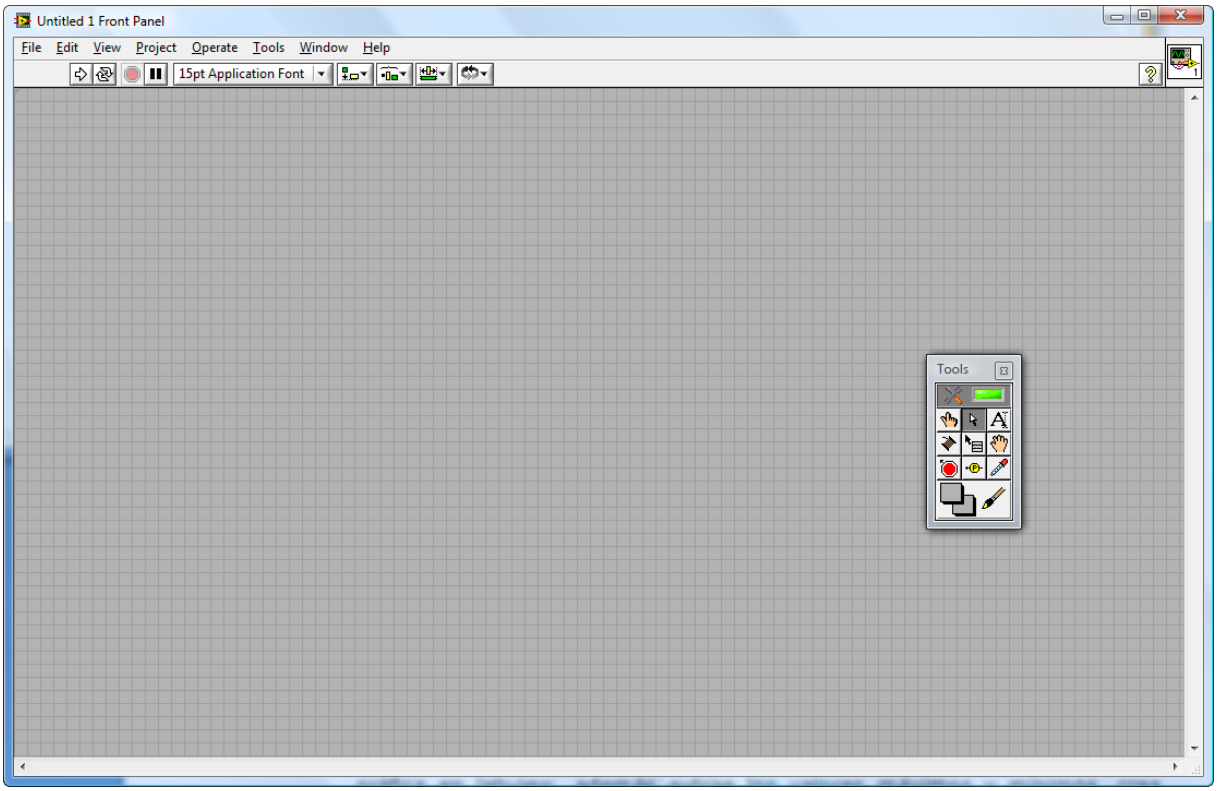


Figura 2. *Panel frontal*

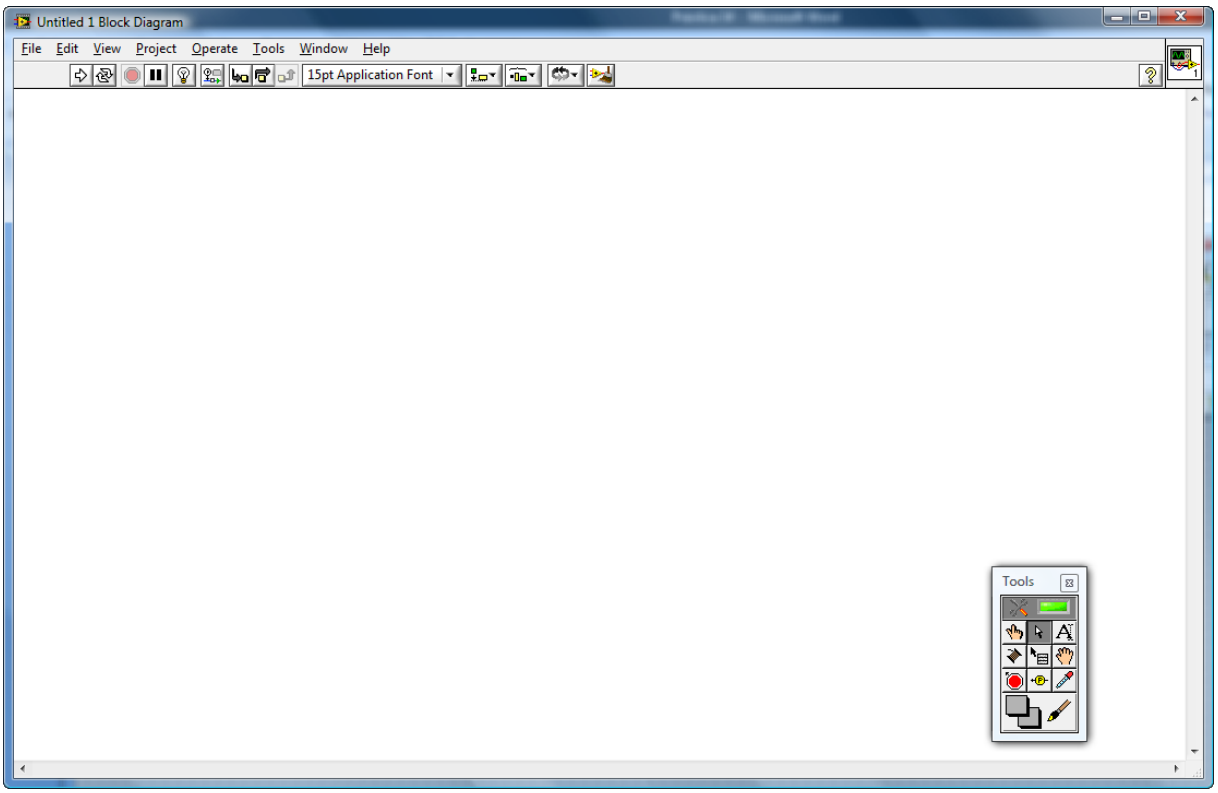


Figura 3. *Diagrama de bloques*

En ambas pantallas podemos encontramos la barra de herramientas "tools" la cual nos permite introducir en nuestro programa todas aquellas funciones que incluye LabVIEW, por ello se debe conocer en detalle las funciones que nos permite el software antes de proceder a la programación. Por lo tanto, se invita al lector a estudiar los numerosos ejemplos que incluye este software o visitar la página web de National Instruments para aprender acerca de las herramientas disponibles, puesto que queda fuera de esta práctica su explicación debido a la gran extensión que esto supondría.

Una vez definido a grandes rasgos en qué consiste este fantástico software de programación, pasaremos a explicar el programa realizado para la comunicación con el PIC16f877A.

El programa toma una medida del microcontrolador y realiza una gráfica en LabVIEW, además, extrae los valores máximos y mínimos, crea una tabla de valores y advierte en caso de superar una consigna establecida. Una de las características principales es que se ha utilizado el control de flujo por hardware, un método prácticamente desconocido, pero muy útil en aplicaciones de este tipo y que viene bien conocer.

Esta práctica se ha dedicado a la realización de un software de supervisión y no de control, por lo tanto el software será más similar a un osciloscopio digital que no a una pantalla de control. Por ello se incluyen como mejoras todos aquellos aspectos sobre control que no se incluyen en esta práctica pero que si se tratan en la siguiente mediante Visual Basic.

Las principales características de este programa son:

- Libertad de elección de comunicación (Puerto, velocidad, …)
- Muestra y guarda valores en tabla.
- Muestra el valor medida de forma analógica, numérica y gráfica.
- Permite guardar la gráfica de la señal en un archivo.
- Calcula y dibuja valores máximo y mínimo.
- Muestra posibles errores en comunicación.
- Dispone de menú de ayuda.
- Advertencia al superar una consigna establecida.

Una vez mostrado todo aquello que es capaz de hacer el programa, pasaremos a indicar todo aquello que no se ha implementado y que se podría añadir como posibles mejoras:

- Comunicación bidireccional que permita enviar el valor de consigna al microcontrolador, para realizar un control en lazo cerrado e implementar algoritmos de control todo/nada, PID.
- Medida de más variables analógicas.
- Visualización/control de señales del tipo digital, interruptores, relés.

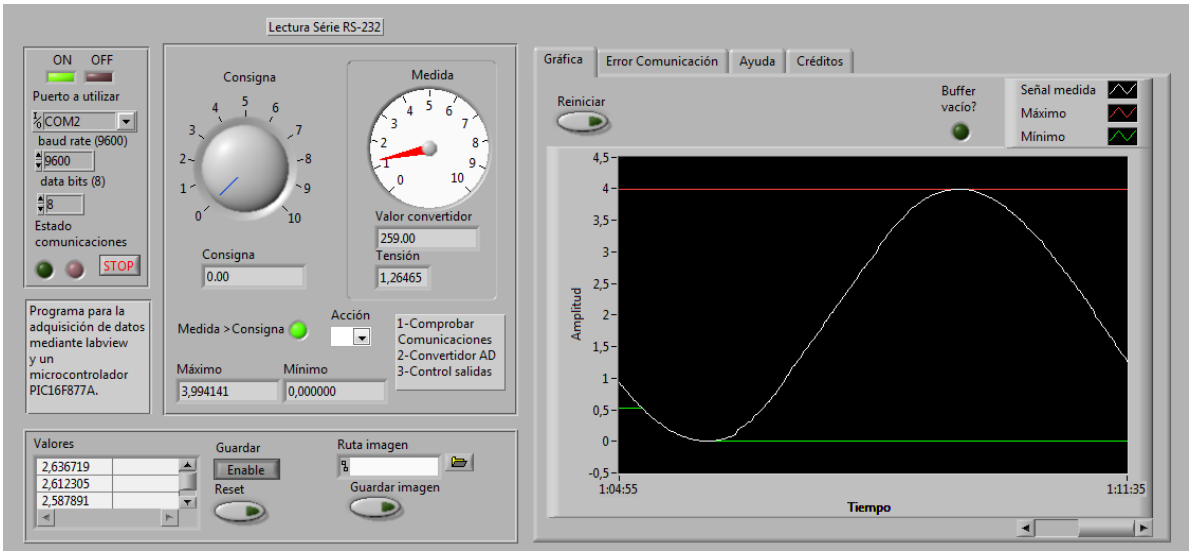
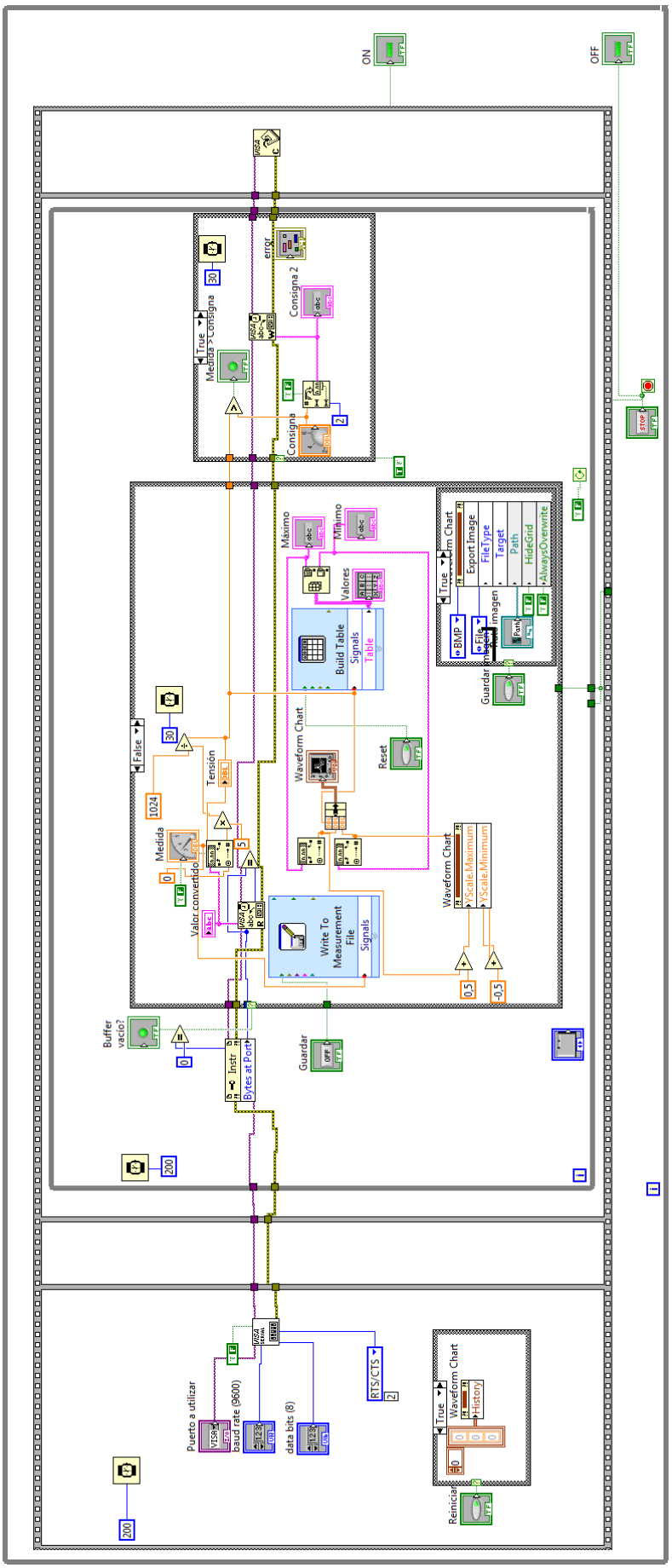


Figura 4. *Panel frontal en funcionamiento*

A continuación se muestra el código del programa, donde podemos observar que primeramente se inicia la comunicación mediante los bloques correspondientes, indicando el puerto, la velocidad y los bits de datos, posteriormente se procede a la lectura del puerto y al procesado de el valor leído para graficarlo y obtener valores máximos.

La estructura del programa se basa en un bucle continuo que tiene anidado 4 "frames" o escenas entre las cuales irá transcurriendo el programa. Se utiliza esta estructura porque garantiza que no pasará a la escena siguiente sin antes acabar todas las operaciones de la actual. El primer cuadro inicializa la comunicación, el segundo está vacío y está dispuesto para ampliación o por si fuera necesario añadir algún retraso en algún tipo de comunicación. El tercer cuadro es el de lectura y procesado de señal, el cúal está formado por un bucle while con dos bloques condicionales en su interior. El primero de éstos detecta si hay un valor en el puerto procediendo a su procesado en caso afirmativo o

saltando este paso en caso contrario. El segundo bloque condicional está gobernado por una constante que permite realizar pruebas de comunicación activando o desactivando éste. En una comunicación normal esta constante debe estar en "True" para que se ejecuten las operaciones correspondientes.. El bucle while se encuentra desactivado por programa puesto que en este código se abre y se cierra el puerto en cada ciclo de comunicación. Por último se cierra la comunicación en la última escena.


ON
OFF
error
Consigna 2
Consigna
Media x Consigna
Máximo
Mínimo
Valores
Build Table
Signals
Table
Waveform Chart
Export Image
FileType
Target
Path
HideGrid
AlwaysOverwrite
BMP
Imagen
Guardar
Reset
Tensión
1024
Medida
Valor convertido
Write To Measurement File
Signals
Waveform Chart
yScale.Maximum
yScale.Minimum
0.5
-0.5
Buffer vacío?
Instr
Bytes at Port
Guardar
200
Puerto a utilizar
baud rate (9600)
data bits (8)
RTS/CTS
Waveform Chart
History
Reinicio

ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA INDUSTRIAL DE BARCELONA

MANUAL DE PRÁCTICAS
PROYECTO FINAL DE CARRERA

Diseño de una plataforma docente para el aprendizaje de microcontroladores 'PIC' de Microchip

PRÁCTICA 15 – Control supervisado Visual Basic 6

Víctor Bueno Álvez

Setiembre 2010

1. OBJETIVOS

En esta práctica, como complemento a la práctica anterior, se continuará trabajando con la comunicación serie, pero esta vez con un programa diferente, el Visual Basic 6. De esta forma conseguimos estudiar las características de ambos programas para un mismo fin, la adquisición de datos mediante un microcontrolador y la visualización de éstos en un ordenador.

2. MATERIAL

Para la realización de esta práctica necesitaremos:

- Placa entrenadora PIC-vBoard.
- Cable de alimentación.
- Programador ICSP PICkit2 o similar.
- Ordenador personal con el programa compilador CCS y el depurador PICkit2.
- Software de desarrollo Visual Basic 6.
- Cable comunicación RS232.

3. DESARROLLO DE LA PRÁCTICA

a) Introducción

Visual Basic es un lenguaje de programación que proviene del BASIC (Beginner's All-purpose Symbolic Instrction Code) y fue desarrollado por Alan Cooper para Microsoft.

La aplicación Visual Basic (Visual Studio) constituye un IDE (Integrated Development Enviroment), es decir, un entorno de desarrollo integrado en el que se incluyen un editor de código para escribir el código fuente, un depurador para corregir posibles errores, un compilador para traducir el código fuente a lenguaje máquina y un constructor de interfaz gráfica o GUI, que nos permite manejar el aspecto gráfico del programa de forma visual y sin necesidad de programación.

b) Programa para el microcontrolador

A continuación se muestra el programa diseñado para el microcontrolador.

En la cabecera de este código se inicializan los diferentes registros del PIC, el módulo LCD, la comunicación I2C con el convertidor analógico digital y la comunicación serie.

De esta forma el sistema queda inicializado y el PIC queda a la espera de recibir algún dato por el puerto serie.

El funcionamiento de la comunicación es sencillo, pero suficiente para la aplicación que se pretende realizar, el programa de Visual Basic enviará un carácter cada vez que se solicite una acción al PIC, entonces éste, al detectar actividad en el puerto lee ese carácter y en función de su valor realiza una tarea u otra. En este caso se han dispuesto 14 tareas diferentes que se son las que se muestra a continuación:

1: Medir temperatura LDR.

2: Aumentar DC led alta luminosidad.

3: Encender resistencia calefactora.

4: Medir temperatura LM35.

5: Disminuir DC led alta luminosidad.

6: Apagar resistencia calefactora.

7: DC led alta luminosidad máximo.

8: DC led alta luminosidad mínimo.

A: Activar relé.

B: Desactivar relé.

M: Medir temperatura tarjeta externa.

N: Encender calefactor tarjeta externa.

O: Apagar calefactor tarjeta externa.

P: Control descentralizado en placa.

```
1: //************************************************************************
2: //   Programa 15.1: Comunicacion serie Visual Basic
3: //   Colección de Prácticas PFC Desarrollo plataforma Docente PIC16F877A
4: //   Autor: Víctor Bueno Álvez
5: //
6: //   Este programa nos permite realizar la supervisión y el control de la
7: //   tarjeta entrenadora PIC-vBoard mediante una comunicación bidireccional
8: //   utilizando el puerto de comunicaciones serie.
9: //   Microcontrolador: PIC16F877A
10: //   Frecuencia: 4 MHZ
11: //   Versión: 1.0
12: //************************************************************************
13:
14: //************************Zona de Definiciones*****************************
15:
16: #include <16f877a.h>
17: #device adc=10 //Usa resolución de 10 bits
18: #fuses hs,nowdt,noprotect,nolvp
19: #use delay(clock=12000000)
20: #use rs232(baud=9600, xmit=pin_c6, rcv=pin_c7,bits=8)
21: #use i2c(master,sda=PIN_C4, scl=PIN_C3) // Para el convertidot PCF8591
22: #BYTE TRISC=0x87
23: #BYTE PORTC=0x07
24: #BYTE TRISA=0x85
25: #BYTE PORTA=0x05
26: #BYTE TRISE=0x89
27: #BYTE PORTE=0x09
28:
29: #include "lcd.c"
30: #define use_portb_kbd TRUE;
31: #include "kbd.c"
32:
33:
34: char valor;
35: char ONOFF='0';
36: char k=0;
37: int x,consigna;
38: int j=0;
39: int muestras=0;
40: int numero[2];
41: float senal[5];
42: float medicion,Recibidos,medicion2,Recibidos2,tempext;
43: int cont,cont3,i;
44: int   ciclo=0;
45:
46: #int_RDA
47:
48: void Serial_isr()
49: {
50: valor=getchar();
51: cont=1;
52: cont3=1;
53: }
54:
55: //************************Zona de Programa*****************************
56:
57: void main ()
58: {
59: set_tris_e(0b00000001);
60:
61: enable_interrupts(INT_RDA);
62: enable_interrupts(GLOBAL);
63:
64: lcd_init();
65: kbd_init();
66: port_b_pullups(TRUE);
67: setup_adc_ports(AN0);
```

```
 68: setup_adc(ADC_CLOCK_INTERNAL);
 69: setup_timer_2(T2_DIV_BY_1,255,1);
 70: bit_clear(TRISC,2);
 71: setup_CCP1(CCP_PWM);
 72:
 73:
 74:    i2c_start();           // Inicio la comunicación I2C
 75:    i2c_write(0b10011110); // Envío Dirección I2C del PCF8591
 76:    i2c_write(0b00000000); // Envío Configuración del PCF8591 para Conv. AD
 77:    i2c_stop();            // Finaliza la comunicación I2C
 78:
 79:    //Descartamos el primer valor medido puesto que siempre es erróneo
 80:    i2c_start();           // Inicio la comunicación I2C
 81:    i2c_write(0b10011111); // Envío Dirección I2C del PCF8591
 82:    Recibidos2=i2c_read();
 83:
 84:
 85: while(true)
 86: {
 87: lcd_putc("\f");
 88:
 89: if((valor)=='1') //Medida Luminosidad LDR
 90: {
 91:    set_adc_channel (1);
 92:    delay_us (20);
 93:    medicion2=read_adc ();
 94:    Recibidos2=(medicion2);
 95:    printf(lcd_putc,"luminosidad %04u ", Recibidos2 );
 96:    if (cont==1){
 97:       printf("%04u", Recibidos2 );
 98:    }
 99:    cont=0;
100:    delay_ms(200);
101: }
102: if((valor)=='2') //Encender Lampara
103: {
104:    if(cont==1){
105:       ciclo++;
106:       set_pwm1_duty(ciclo);
107:       cont=0;
108:    }
109:    lcd_putc("Lampara:Encendiendo\n");
110:    printf(lcd_putc,"Nivel: %03u/255",ciclo);
111:    delay_ms(200);
112: }
113:
114: if((valor)=='3') //Encender Calefactor
115: {
116:    output_high(pin_E0);
117:    lcd_putc("Calefactor:\n");
118:    lcd_putc("Encendido");
119:    delay_ms(200);
120: }
121:
122:
123: if((valor)=='4') // Mide temperatura "LM35"
124: {
125:    set_adc_channel (0);
126:    delay_us (20);
127:    medicion=read_adc ();
128:    Recibidos=(medicion*(0.48875));
129:    printf(lcd_putc,"temperatura %02.1f ", Recibidos );
130: if (cont==1)
131: {
132:    printf("%02.1f", Recibidos );
133: }
134:    cont=0;
```

```
135:    delay_ms(200);
136: }
137:
138: if((valor)=='5') //Apagar lampara
139: {
140:    if(cont==1)
141:    {
142:       ciclo--;
143:       set_pwm1_duty(ciclo);
144:       cont=0;
145:    }
146:    lcd_putc("Lampara:Apagando\n");
147:    printf(lcd_putc,"Nivel: %03u/255",ciclo);
148:    delay_ms(200);
149: }
150: if((valor)=='6') //Apagar calefactor
151: {
152:    output_low(pin_E0);
153:    lcd_putc("Calefactor:  \n");
154:    lcd_putc("Apagado   ");
155:    delay_ms(100);
156: }
157:
158: if((valor)=='7') //Encender lámpara máximo
159: {
160:    set_pwm1_duty(255);
161:    lcd_putc("Lampara:Encendida\n");
162:    printf(lcd_putc,"Nivel: 255");
163:    delay_ms(200);
164:    }
165:
166: if((valor)=='8') //Apagar lámpara
167: {
168:    set_pwm1_duty(0);
169:    lcd_putc("Lampara:Apagada\n");
170:    printf(lcd_putc,"Nivel: 0");
171:    delay_ms(200);
172: }
173:
174: if((valor)=='A') //Encender Relé
175: {
176:    bit_set(PORTE,2);
177:    lcd_putc("Rele: \n");
178:    lcd_putc("Encendido    ");
179:    delay_ms(100);
180: }
181:
182: if((valor)=='B') //Desactivar Relé
183: {
184:    bit_clear(PORTE,2);
185:    lcd_putc("Rele: \n");
186:    lcd_putc("Apagado     ");
187:    delay_ms(100);
188:    }
189:
190: if((valor)=='M') //Medida temperatura tarjeta externa
191: {
192:    Recibidos2=i2c_read();
193:    printf("%02.1f",(Recibidos2/10)+10 );
194:    delay_ms(200);
195: }
196:
197: if((valor)=='N') //Encender calefactor tarjeta externa
198: {
199:    bit_set(PORTE,1);
200:    Recibidos2=i2c_read();
201:    printf("%02.1f",(Recibidos2/10)+10 );
```

```
202:     delay_ms(200);
203:     }
204:
205: if((valor)=='O') //Desactivar calefactor tarjeta externa
206: {
207:     bit_clear(PORTE,1);
208:     Recibidos2=i2c_read();
209:     printf("%02.1f",(Recibidos2/10)+10 );
210:     delay_ms(200);
211:     }
212:
213: if((valor)=='P') //Control en la placa
214: {
215:     lcd_putc('\f');
216:     printf(lcd_putc,"Introd. consigna:\n");
217:     printf(lcd_putc,"Consigna:");
218:     delay_ms(200);
219:
220:     i=0;
221:     while(i<=2){
222:        if (k!=0){
223:
224:           if (k=='9'){
225:           printf(lcd_putc,"\f Introd. consigna:\n");
226:           printf(lcd_putc,"Consigna:");
227:           delay_ms(20);
228:           i=0;
229:           }
230:           else{
231:           lcd_putc(k);
232:           numero[i]=k-48;
233:           delay_ms(200);
234:           i++;
235:           }
236:       }
237:       k=kbd_getc();
238: }
239:
240:     consigna=(numero[1]*10)+numero[2];
241: while((valor)=='P'){
242:     tempext=i2c_read();
243:     senal[j]=tempext;
244:     j=j+1;
245:     printf(lcd_putc,"\fConsig.: %2u    %c\n", consigna,ONOFF);
246:     printf(lcd_putc,"Temper.: %02.1f ",(tempext/10)+10 );
247:     delay_ms(500);
248:     if(consigna<((tempext/10)+10 )){
249:        ONOFF='0';
250:        bit_clear(PORTE,1);
251:        }
252:     else{
253:        ONOFF='1';
254:        bit_set(PORTE,1);
255:        }
256:     delay_ms(500);
257:     }
258:
259: }
260: }
261: }
262:
```

c) Programa Visual Basic 6

Ahora procederemos a una explicación sobre el funcionamiento del programa Visual Basic 6 para posteriormente pasar a a explicar las funciones del programa diseñado.

Lo primero que nos encontramos al ejecutar el programa Visual Basic 6 es un menú de nuevo proyecto el cual nos permite seleccionar entre distintos tipos de aplicaciones. En nuestro caso, como en la mayoría de los casos, seleccionaremos EXE estándar.

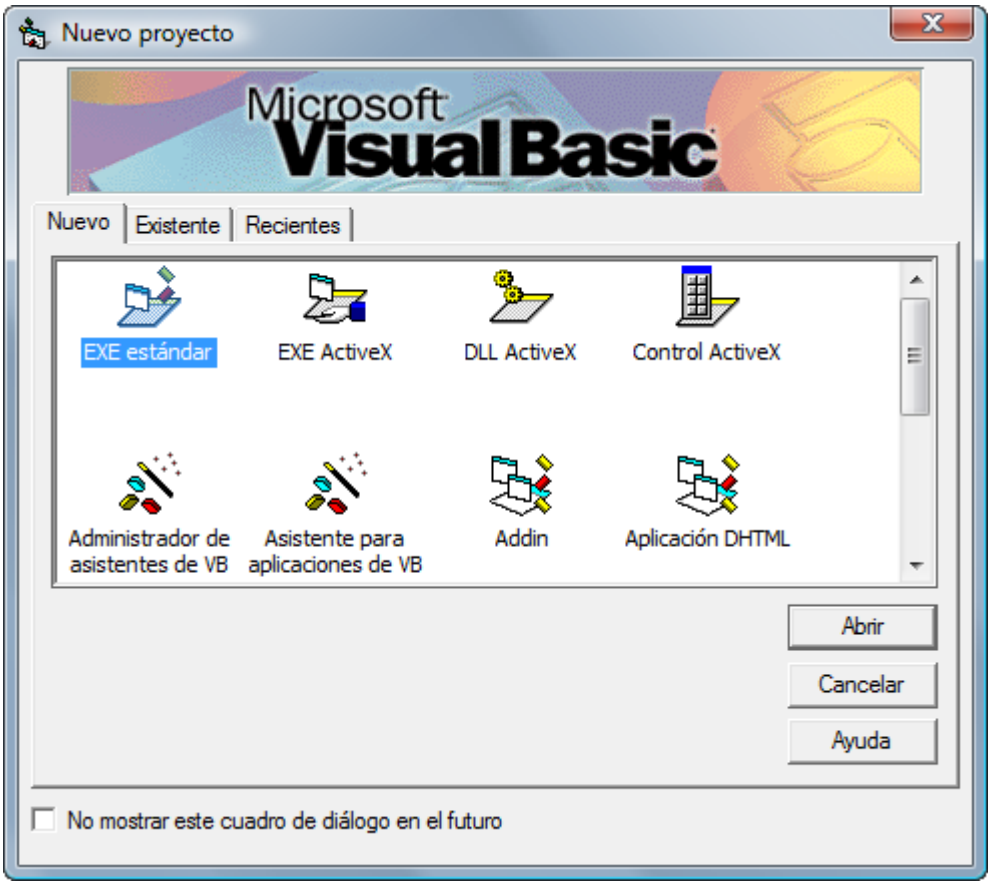


Figura 1. Pantalla inicio

Una vez creado el nuevo proyecto nos aparecerá una nueva ventana que será sobre la cual diseñaremos nuestro programa. Debemos saber que la programación en Visual Basic se basa en la colocación de objetos sobre formularios que formarán la parte visual, y la programación de estos objetos para implementar la parte de cálculo. Estos formularios constituirán las distintas pantallas de nuestro programa, entre las cuales podremos navegar e intercambiar variables.

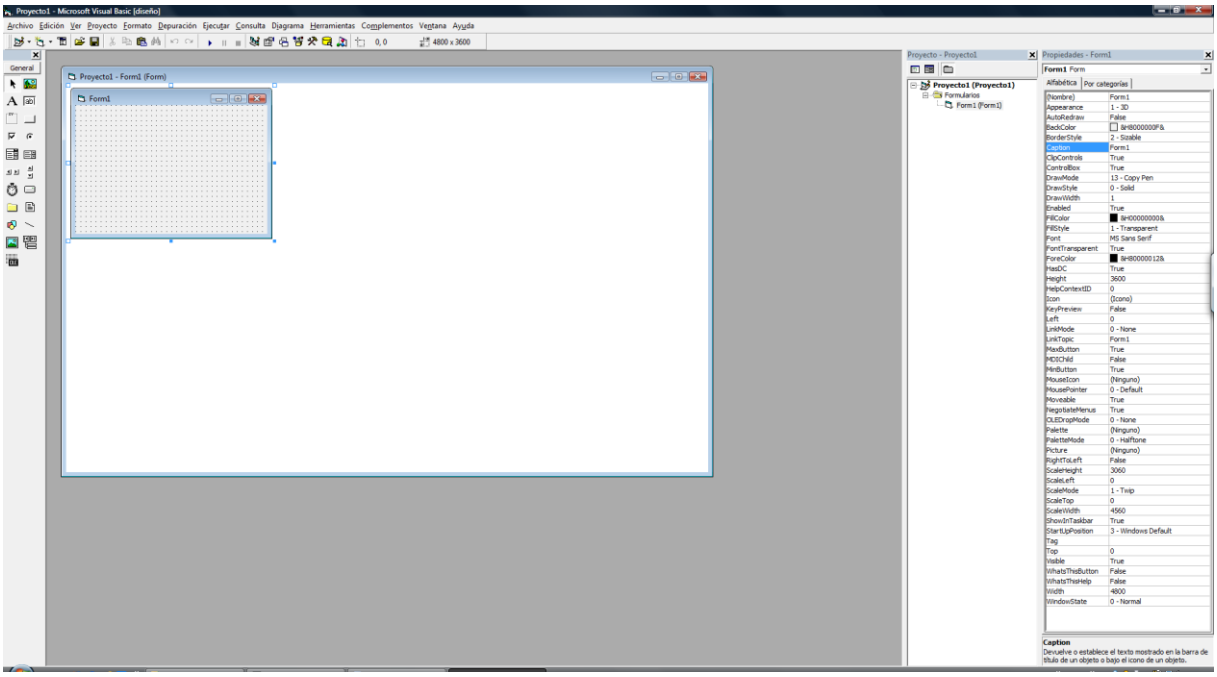


Figura 2. *Pantalla de trabajo*

En la pantalla de trabajo se pueden diferenciar, a grandes rasgos, 5 zonas principales de trabajo.

La primera, situada en la parte superior de la pantalla consiste en una barra de herramientas similar a las que estamos acostumbrados a utilizar en otros programas de Windows, con ella podemos abrir y guardar archivos, así como configurar nuestro proyecto.

La segunda zona, situada a la izquierda, corresponde con una barra de herramientas donde encontraremos todas las herramientas que nos ofrece Visual Basic para incluir en nuestro programa. En la figura anterior se observan aquellas que vienen activadas por defecto, como son los botones, cuadros de texto, figuras, etc.., pero existen una infinidad de herramientas añadidas que se pueden incorporar haciendo clic con el botón derecho del ratón.

La tercera zona, situada a la derecha, está formada por la barra de proyecto, donde se muestra el desglose de nuestro programa en sus distintos formularios, y la barra de propiedades, en donde se muestran las características de los elementos del formulario y desde donde podemos variar las funciones de todos los elementos sin recurrir a la programación.

Por último, las zonas 4 y 5, son las zonas de trabajo, formadas por la vista de formulario y la pantalla de programación. La primera, de color gris, nos muestra el aspecto visual de nuestro formulario y nos permite acceder a la pantalla de programación haciendo doble clic sobre el elemento que queremos programar.

Una vez definido el funcionamiento general del programa, se invita al lector a investigar acerca de las diferentes opciones que ofrece Visual Basic, ya sea mediante los programas de prueba que trae el software o con la ayuda de la infinidad de ejemplos que se encuentran en la web. Además, se anima al

lector a realizar su primer proyecto sencillo y a generar el archivo *.exe correspondiente, para que pueda observar lo sencillo que puede llegar a ser la realización de un programa mediante un lenguaje tan potente como es el Visual Basic.

A continuación se definirán los aspectos principales del programa diseñado para la comunicación con la tarjeta entrenadora PIC –vBoard y se mostrará el código del programa resultante.

El programa realizado en Visual Basic se comunica con el PIC mediante comunicación RS-232 utilizando control de flujo por software, y nos permite visualizar las variables de la planta a controlar así como actuar sobre dispositivos conectados a sus salidas, con tal de mantener los niveles deseados.

Como magnitudes de entrada podemos encontrar:

- 2 sensores de temperatura, 1 para cada estancia.
- 1 sensor de luminosidad

Además podemos encontrar acciones de control como:

- Control de lámpara por PWM para escoger el valor exacto de iluminación.
- 2 calefactores, 1 para cada estancia.
- 1 Relé controlable manualmente para apertura de puerta, sirena de emergencia, etc…

Estos son los periféricos conectados directamente al PIC, aunque podrían ser ampliados fácilmente se consideran suficientes para realizar un programa de carácter didáctico. Además, utilizando las diversas opciones que nos proporciona el programa Visual Basic y los microcontroladores PIC podemos diseñar un programa con un nivel de complejidad bastante elevado, pudiendo variar el tipo de control, los parámetros, etc…

Las principales características de este programa són:

- Tipos de control:
    a. Automático
    b. Manual
    c. En placa

- Medidas manuales.
    a. Luminosidad
    b. Temperatura
- Acciones manuales.
    a. Lámpara ON/OFF y Incremento/Decremento secuencial iluminación.
    b. Calefactor 1 ON/OFF.
    c. Calefactor 2 ON/OFF.
    d. Relé ON/OFF.
- Barras de nivel para las 3 magnitudes.
- Gráfica para control automático.
- Exportación imagen gráfica como BMP.
- Menú ficheros para guardar los datos en Excel.
- Posibilidad de seleccionar cualquier puerto COM.
- Visualización Reloj-Calendario.
- Menú Ayuda.

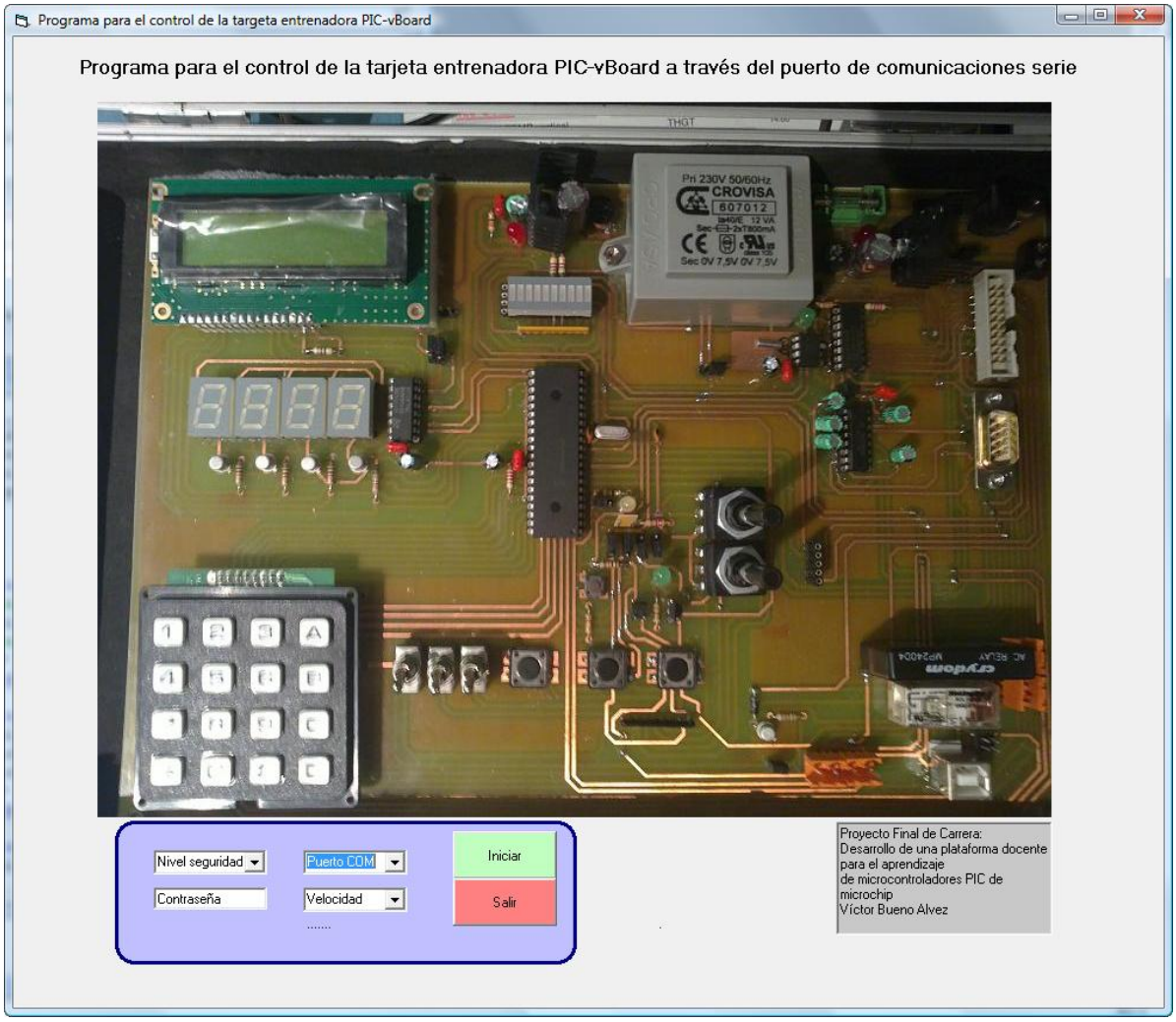


Figura 3. *Pantalla Inicio*

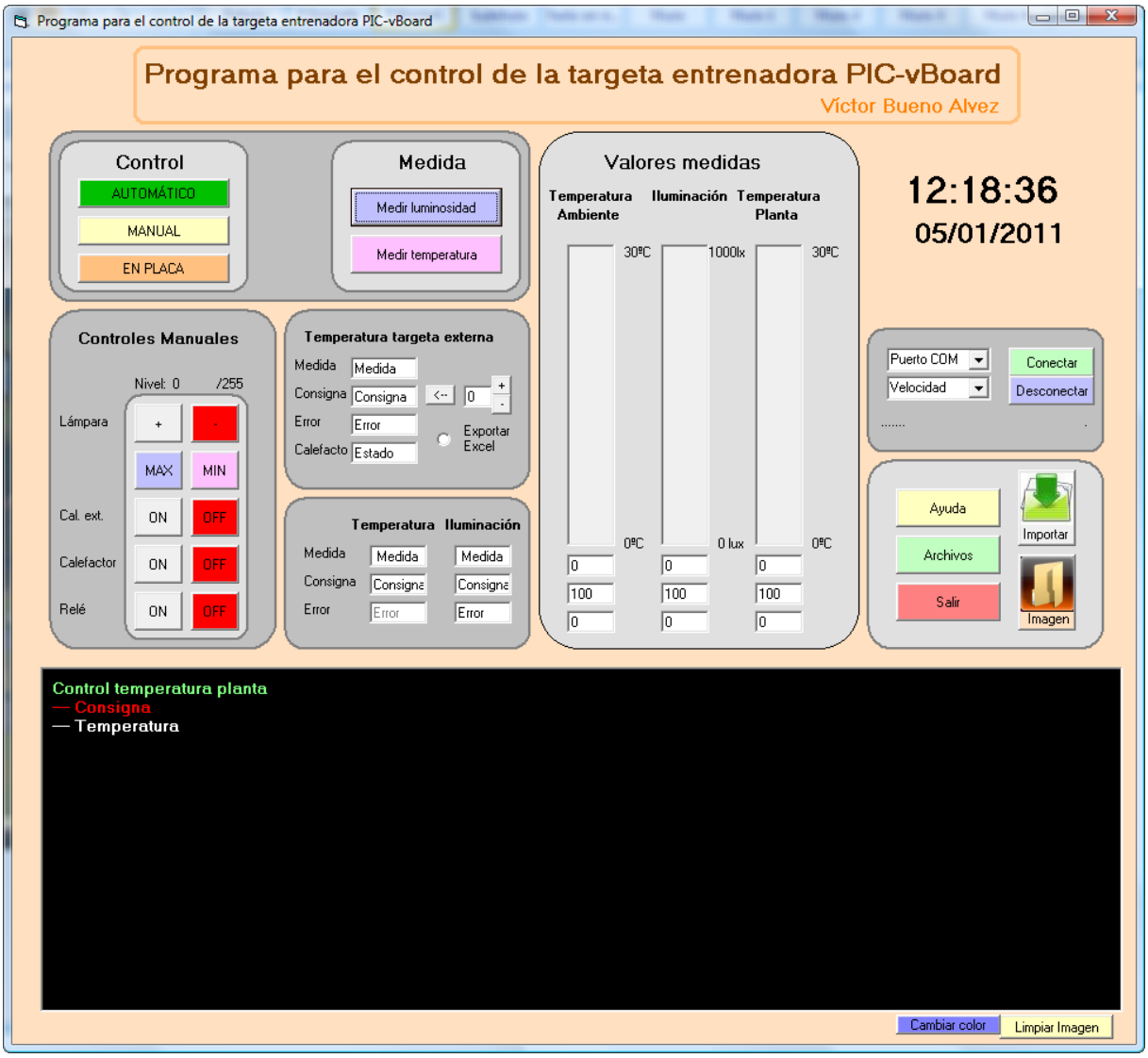


Figura 4. *Pantalla Principal*

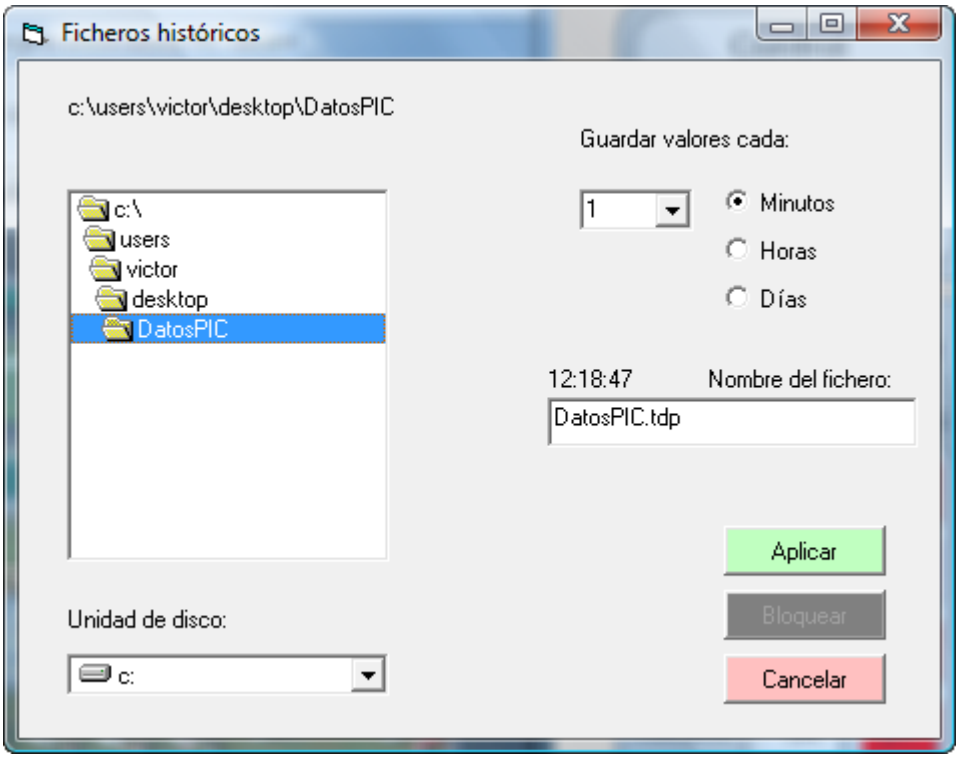


Figura 5. *Menú ficheros*

En el programa se han intentado implementar todas aquellas funciones que se consideran útiles en un ámbito didáctico y las cuales pueden ser implementadas con el microcontrolador PIC16F877A, pero siempre cabe la opción de escoger otro microcontrolador o diseñar una nueva aplicación para otro tipo de control, con lo cual se pueden definir una serie de posibles mejoras:

- Medida de tensión en los potenciómetros.
- Control de la salida analógica PCF8591
- Medida 3 entradas analógicas PCF8591
- Visualización estado 3 pulsadores.
- Comunicación USB.
- Control PID.

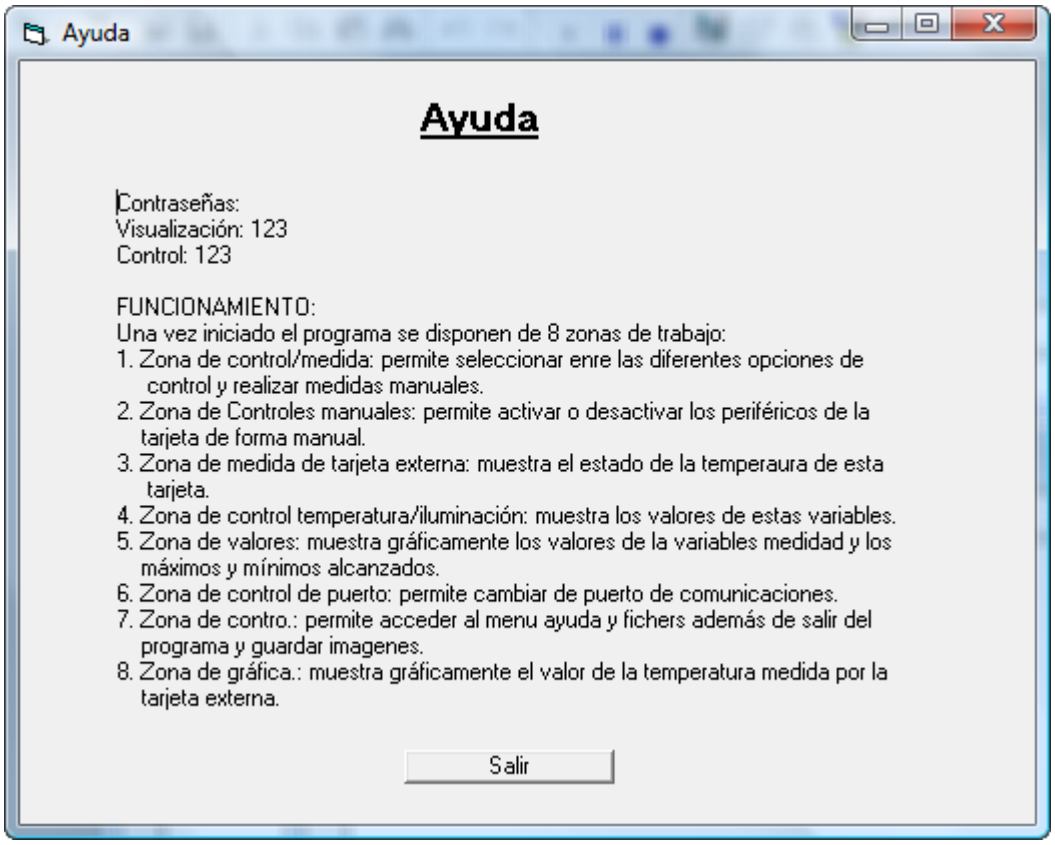


Figura 6. *Pantalla ayuda*

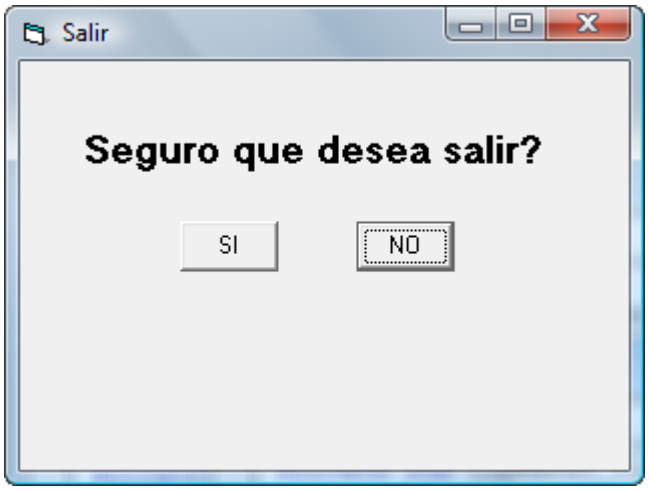


Figura 7. *Pantalla salir*

```
Form1 - 1

Dim InBuff As String
Dim medida, consigna2, error, flag, flag2, flagmicro, valorx, zoom, muestrasi, muestrast, m
uestrastext, puerto, muestra As Integer
Dim valor As Integer
Dim Lamp, calef, rele, control As String



Private Sub Command1_Click()    'medida luminosidad
MSComm1.Output = "1" 'Envia el numero "1" por el puerto (mide temperatura)
flag = 0
End Sub

Private Sub Command10_Click() 'relé ON
MSComm1.Output = "A" 'Envia el numero "A" por el puerto
Command10.BackColor = &HFF00&
Command11.BackColor = &H8000000F
rele = "ON"
End Sub

Private Sub Command11_Click()   'Relé OFF
MSComm1.Output = "B" 'Envia el numero "2" por el puerto
Command10.BackColor = &H8000000F
Command11.BackColor = &HFF&
rele = "OFF"
End Sub

Private Sub Command12_Click()   'Abré menú archivos
Form3.Show
End Sub

Private Sub Command13_Click()
MSComm1.PortOpen = False
Form4.Show
Form1.Hide
End Sub

Private Sub Command14_Click()   'calefactor externo ON
 MSComm1.Output = "N" 'Envia el numero "N" por el puerto
 Text12.Text = "ENCENDIDO"
 Command14.BackColor = &HFF00&
Command15.BackColor = &H8000000F
End Sub

Private Sub Command15_Click()   'calefactor externo OFF
 MSComm1.Output = "O" 'Envia el numero "N" por el puerto
 Text12.Text = "apagado"
 Command14.BackColor = &H8000000F
Command15.BackColor = &HFF&
End Sub

Private Sub Command16_Click()   'acepta consigna
Text10.Text = Text8.Text
consigna2 = Text10.Text

End Sub

Private Sub Command17_Click()   'aumenta consigna
Text8.Text = Text8.Text + 0.1
End Sub

Private Sub Command18_Click()   'disminuye coonsigna
Text8.Text = Text8.Text - 0.1
End Sub

Private Sub Command19_Click()
'x = 0
flag = 3
MSComm1.Output = "Q" 'Envia el numero "N" por el puerto
```

```
Form1 - 2

End Sub

Private Sub Command2_Click()    'lampara +
MSComm1.Output = "2" 'Envia el numero "2" por el puerto
Command2.BackColor = &HFF00&
Command5.BackColor = &H8000000F
Text7.Text = Val(Text7.Text) + 1
Lamp = "ON"

'Shape3.Count.Visible = False
End Sub

Private Sub Command20_Click()
Picture1.Picture = LoadPicture("")
x = 0

End Sub

Private Sub Command21_Click()   'control en placa
MSComm1.Output = "P" 'Envia el numero "P" por el puerto
End Sub

Private Sub Command22_Click()   'Guarda imagen

 Dim Imagen As IPictureDisp
 Set Imagen = Picture1.Image

SavePicture Imagen, "C:\Users\Victor\Desktop\datosPIC\Imagen.BMP"

End Sub

Private Sub Command23_Click()   'Conecta al puerto
If MSComm1.PortOpen = False Then
    With MSComm1
    .CommPort = Val(Combo1.Text) 'Numero del puerto
    .Settings = "9600,N,8,1" 'Rata de baudios, paridad, bits de datos, bits de parada
    .Handshaking = comNone
    .InputMode = comInputModeText
    .RThreshold = 4 'Define cada cuantos bytes recibidos se genera un evento
    .InputLen = 4 'Cuantos bytes se extraen al leer el puerto
    .PortOpen = True
    End With
    Else
    Label2.Caption = "Desconecte entrenadora e intente de nuevo"
End If

Label23.Caption = "Conectado correctamente al puerto:"
Label24.Caption = Combo1.Text
End Sub

Private Sub Command24_Click()
If (Picture1.BackColor = &HFFFFFF) Then
    Picture1.BackColor = &H0&
Else
    Picture1.BackColor = &HFFFFFF
End If
End Sub

Private Sub Command25_Click()   'Control manual
MSComm1.Output = "8" 'Envia el numero "8" por el puerto
Command7.BackColor = &H8000000F
Command8.BackColor = &HFF&
Command2.Enabled = True
Command3.Enabled = True
Command5.Enabled = True
Command6.Enabled = True
Command1.Enabled = True
Command4.Enabled = True
Timer1.Enabled = False
control = "manual"
```

```
Form1 - 3

End Sub

Private Sub Command26_Click()   'lampara OFF
MSComm1.Output = "8" 'Envia el numero "8" por el puerto
Text7.Text = "0"

End Sub

Private Sub Command27_Click() 'lampara ON
MSComm1.Output = "7" 'Envia el numero "7"
Text7.Text = "255"

End Sub

Private Sub Command29_Click()
MSComm1.Output = "T" 'Envia el numero "1" por el puerto (mide temperatura)
'MSComm1.PortOpen = False
flag = 3
End Sub

Private Sub Command28_Click()   'Abre menú ayuda
Form1.Hide
Form4.Show
End Sub

Private Sub Command3_Click()    'calefactor ON
MSComm1.Output = "3" 'Envia el numero "3" por el puerto
Command3.BackColor = &HFF00&
Command6.BackColor = &H8000000F
calef = "ON"
End Sub

Private Sub Command4_Click()    'medida temperatura
MSComm1.Output = "4" 'Envia el numero "4" por el puerto(medir temperatura)
flag = 1
End Sub

Private Sub Command5_Click()    'lampara -
MSComm1.Output = "5" 'Envia el numero "5" por el puerto
Command2.BackColor = &H8000000F
Command5.BackColor = &HFF&
Text7.Text = Val(Text7.Text) - 1
Lamp = "OFF"
End Sub

Private Sub Command6_Click()    'calefactor externo OFF
MSComm1.Output = "6" 'Envia el numero "6" por el puerto
Command3.BackColor = &H8000000F
Command6.BackColor = &HFF&
Lamp = "OFF"
End Sub

Private Sub Command7_Click() 'Control automático

If (((Val(Text9.Text) - Val(Text10.Text)) > 0.5)) Then
       MSComm1.Output = "O" 'Envia el numero "O" por el puerto
        Text12.Text = "apagado"
        flagmicro = 1

    ElseIf (((Val(Text9.Text) - (Val(Text10.Text))) < -0.5)) Then
        MSComm1.Output = "N" 'Envia el numero "N" por el puerto
        Text12.Text = "encendido"
        flagmicro = 0
   End If

Timer1.Enabled = True
flag = 2
Command7.BackColor = &HFF00&
Command8.BackColor = &H8000000F
Command1.Enabled = False
```

```
Form1 - 4

Command4.Enabled = False
control = "automático"

End Sub

Private Sub Command8_Click()    'Desconecta del puerto

End Sub

Private Sub Command9_Click()    'Sale del programa
Form1.Hide
Form5.Show
End Sub

Private Sub Form_Load()

flag2 = 0
valorx = 0
flagmicro = 0
consigna2 = 0
zoom = 1
flag = 6

Lamp = "OFF"
calef = "OFF"
rele = "OFF"
control = "OFF"
muestrasi = 0
muestrast = 0
muestrastext = 0
muestra = 0

End Sub

Private Sub MSComm1_OnComm()
InBuff = MSComm1.Input

If (flag = 0) Then      'iluminación
    If Val(InBuff) <> 0 Then
    Text2.Text = InBuff
    medida = CInt(Val(InBuuf))
    Text6.Text = Val(Text2.Text) - Val(Text3.Text)
    Text6.Text = Format(Text6.Text, "#0#")
    Shape6.Top = 5000 - (5 * Val(InBuff)) ''''''''''''''''''''''''''''
    ProgressBar2.Value = Val(Text2.Text)
    Text14.Text = Val(Text2.Text)

    If Val(Text2.Text) > Val(Text19.Text) Then
        Text19.Text = InBuff
    ElseIf Val(Text2.Text) < Val(Text20.Text) Then
        Text20.Text = InBuff
    End If
    End If

ElseIf (flag = 1) Then   'temperatura
If Val(InBuff) <> 0 Then
    Text1.Text = InBuff
    medida = CInt(Val(InBuuf))
    Text4.Text = Val(Text1.Text) - Val(Text5.Text)
    Text4.Text = Format(Text4.Text, "#0.0#")
    Shape5.Top = 5000 - (130 * Val(InBuff))
    ProgressBar1.Value = Val(Text1.Text)
    Text13.Text = Val(Text1.Text)


    If Val(Text1.Text) >= Val(Text16.Text) Then
        Text16.Text = InBuff
    ElseIf Val(Text1.Text) <= Val(Text17.Text) Then
        Text17.Text = InBuff
    End If
```

```
Form1 - 5

    End If

ElseIf (flag = 2) Then      'externa
    If Val(InBuff) <> 0 Then

        Text9.Text = InBuff
        medida = CInt(Val(InBuuf))
        For i = 1 To zoom
        Picture1.PSet (valorx, Picture1.Height - 500 - (150 * (consigna2 - 10))), vbRed
        Picture1.PSet (valorx, Picture1.Height - 500 - (150 * (Val(InBuff) - 10)))
        valorx = valorx + 1
        Next i
        Text11.Text = (Val(Text9.Text) - (Val(Text10.Text)))
        ProgressBar3.Value = Val(Text9.Text)
        Text15.Text = Val(Text9.Text)

        If Val(Text9.Text) > Val(Text22.Text) Then
            Text22.Text = Val(InBuff)
        ElseIf Val(Text9.Text) < Val(Text23.Text) Then
            Text23.Text = Val(InBuff)
        End If

    End If

ElseIf (flag = 3) Then      'iluminación
    Text18.Text = InBuff

End If

End Sub

Private Sub S_Click()
End
End Sub

Private Sub Option1_Validate(Cancel As Boolean)
T = Chr(9)
MiFich = Form3.Dir1.Path & "\" & Form3.Text1.Text 'Form3.NomFichero.Caption '
Open MiFich For Output As #1
Print #1, T & "Temperatura" & T & "Muestra"

End Sub


Private Sub Timer1_Timer()  'Temporizador para controla automático


If (((Val(Text9.Text) - Val(Text10.Text)) > 0)) Then
       MSComm1.Output = "O" 'Envia el numero "2" por el puerto
        Text12.Text = "apagado"
        flagmicro = 1

ElseIf (((Val(Text9.Text) - (Val(Text10.Text))) < -0.5)) Then
        MSComm1.Output = "N" 'Envia el numero "N" por el puerto
        Text12.Text = "encendido"
        flagmicro = 0
End If


End Sub


Private Sub VScroll1_Change()
Text8.Text = Val(Text8.Text) + 0.1

End Sub

Private Sub Timer2_Timer()  'temporizador para guardar fichero
Dim MiFich As String
Dim T As String
```

```
Form1 - 6

Dim Trinicio As Single
T = Chr(9) ' ASCII del tabulador para separar campos y permitir
    ' que se puedan leer los ficheros con extensión *.tdp
        muestra = muestra + 1
                  Print #1, T & Lamp & T & calef & T & rele & T & control & T & Time & T
& Val(Text9.Text) & T & muestra

End Sub

Private Sub Timer3_Timer()  'temporizador para mostrar la hora

Dim Hora As Variant
Hora = Time
Label30.Caption = Date
Label31.Caption = Time

End Sub
```

```
Form2 - 1

Public puerto As Double

Private Sub Command1_Click()


    If Combo2.Text = "Visualización" Then 'aceptacion de la contraseña
        If Text3.Text = "123" Then 'aceptacion de la contraseña
        Form1.Show
        Form2.Hide
        End If

    End If

    If Combo2.Text = "Control" Then 'aceptacion de la contraseña
        If Text3.Text = "123" Then 'aceptacion de la contraseña
        Form1.Show
        Form2.Hide
        End If
    End If

Text3.Text = ""
Label2.Caption = "Clave Incorrecta"


End Sub

Private Sub Command2_Click()

End Sub

Private Sub Command3_Click()
If MSComm1.PortOpen = True Then MSComm1.PortOpen = False
End
End Sub

Private Sub Form_Load()
Const clave = "PIC-vBoard"

End Sub

Private Sub Text1_Change()

End Sub
```

```
Form3 - 1

Private Sub Command1_Click()

T = Chr(9)
MiFich = Form3.Dir1.Path & "\" & Form3.Text1.Text 'Form3.NomFichero.Caption '
Open MiFich For Output As #1
Print #1, T & "Lampara" & T & "Calefactor" & T & "Relé" & T & "Control" & T & "Hora" & T &
T & "Temperatura" & T & "Muestra"
Label2.Caption = "Archivo Abierto"

Command2.Enabled = True

Tinicio = Timer

Form1.Timer2.Enabled = True



End Sub

Private Sub Command2_Click()


Form1.Timer2.Enabled = False
    Close #1
    Label2.Caption = "Archivo Cerrado"
End Sub

Private Sub Command3_Click()

Form3.Hide
Form1.Show

End Sub

Private Sub Command4_Click()



End Sub

Private Sub Command5_Click()

End Sub

Private Sub Dir1_Change()

NomDir.Caption = Dir1.Path

End Sub

'Private Sub Drive1_Change()
 '   On Error GoTo DriverError
  '       Dir1.Path = Drive1.Drive
  '  Exit Sub
'DriveError:
'    MsgBox "Error: unidad no preparada", vbExclamation, "Error"
'    Exit Sub
'End Sub

Private Sub Form_Load()
    Dir1.Path = "c:\users\victor\desktop\DatosPIC"
    NomDir.Caption = Dir1.Path
    dia = Date
    'Tinicio = Timer 'Establece la hora de inicio
    'Timer3.Enabled = False
    Label4.Caption = Time
    Text1.Text = "DatosPIC.tdp"
End Sub
```

```
Form4 - 1

Private Sub Command1_Click()
Form4.Hide
Form1.Show
End Sub
```

```
Form5 - 1

Private Sub Command1_Click()
If MSComm1.PortOpen = True Then MSComm1.PortOpen = False
End
Close #1
End Sub

Private Sub Command2_Click()
Form5.Hide
Form1.Show
End Sub
```